ਜਾਵਾ ਨਾਲ OOP ਦੀ ਜਾਣ-ਪਛਾਣ
(Introduction to OOP with Java)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

4.1 ਓਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਨਾਲ ਜਾਣ-ਪਛਾਣ
4.2 ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਸਿਧਾਂਤ - ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction), ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਇਨਹੇਰੀਟੈਂਸ (Inheritance), ਅਤੇ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism)
4.3 ਜਾਵਾਂ ਨਾਲ ਜਾਣ ਪਛਾਣ
4.4 ਜਾਵਾ ਦਾ ਇਤਿਹਾਸ
4.5 ਜਾਵਾ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ
4.6 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਮੁੱਢਲੀ ਬਣਤਰ
4.7 ਇਕ ਸਾਧਾਰਣ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣਾ ਅਤੇ ਚਲਾਉਣਾ
4.8 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਮੁੱਢਲੇ ਤੱਤ: ਕਰੈਕਟਰ ਸੈੱਟ, ਟੋਕਨਜ਼, ਕਮੈਂਟਸ

4.1 ਆਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਨਾਲ ਜਾਣ-ਪਛਾਣ (INTRODUCTION TO OBJECT ORIENTED PROGRAMMING)

ਪ੍ਰੋਸੀਜ਼ਰਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਡਾਟਾ ਉਪਰ ਵੱਖ-ਵੱਖ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਬਣਾਏ ਜਾਣ ਵਾਲੇ ਪ੍ਰੋਸੀਜ਼ਰਾਂ (procedures) ਜਾਂ ਮੈਥਡਜ਼ (methods) ਨਾਲ ਸੰਬੰਧਤ ਹੈ, ਜਦੋਂ ਕਿ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਆਬਜੈਕਟ (object) ਬਣਾ ਕੇ ਕੰਮ ਕਰਨ ਨਾਲ ਸੰਬੰਧਤ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ **"ਆਬਜੈਕਟ (object)"** ਦੀ ਧਾਰਨਾ 'ਤੇ ਕੰਮ ਕਰਦੀਆਂ ਹਨ। ਆਬਜੈਕਟਸ ਅਸਲ-ਸੰਸਾਰ ਦੀਆਂ ਵਸਤੂਆਂ (entities) ਹੁੰਦੀਆਂ ਹਨ ਜਿਵੇਂ ਕਿ: ਵਿਦਿਆਰਥੀ, ਵਿਅਕਤੀ, ਪੈਨ, ਟੇਬਲ, ਟੈਲੀਵਿਜ਼ਨ, ਕੰਪਿਊਟਰ ਆਦਿ।

ਸੰਸਾਰ ਵਿੱਚ ਕਿਸੇ ਵੀ ਚੀਜ ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ, ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਨੂੰ ਉਸਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (properties) ਅਤੇ ਵਿਵਹਾਰ (behaviour) ਦੇ ਰੂਪ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ: ਜੇਕਰ ਆਬਜੈਕਟ ਇੱਕ ਵਿਅਕਤੀ ਹੈ ਤਾਂ ਉਸਦਾ ਨਾਮ, ਉਮਰ, ਕੱਦ, ਭਾਰ, ਆਦਿ ਉਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੋਣਗੀਆਂ ਅਤੇ ਤੁਰਨਾ, ਬੋਲਣਾ, ਸੌਣਾ ਆਦਿ ਉਸਦਾ ਵਿਵਹਾਰ (ਕਿਰਿਆਵਾਂ) ਹੋ ਸਕਦੇ ਹਨ। ਹਰੇਕ ਆਬਜੈਕਟ/ਵਸਤੂ ਦੀਆਂ ਆਪਣੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਇਹ ਦੱਸਦੀਆਂ ਹਨ ਕਿ ਉਹ ਕੀ ਚੀਜ਼ ਹੈ ਜਾਂ ਉਹ ਕੀ ਕਰਦੀ ਹੈ।

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿਧੀ ਵਿੱਚ, ਕਲਾਸਾਂ ਅਤੇ ਆਬਜੈਕਟਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਵਿਧੀ ਕੁਝ ਸੰਕਲਪਾਂ (concepts) ਜਿਵੇਂ ਕਿ: ਆਬਜੈਕਟ (object), ਕਲਾਸ (class), ਇਨਹੇਰੀਟੈਂਸ (Inheritance), ਪੋਲੀਮੋਰਫਿਜ਼ਮ (polymorphism), ਐਬਸਟਰੈਕਸ਼ਨ (absraction) ਅਤੇ ਐਨਕੈਪਸੂਲੇਸ਼ਨ (encapsulation) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਸਾਫਟਵੇਅਰ ਨੂੰ ਤਿਆਰ ਕਰਨ ਅਤੇ ਉਸਦੇ ਰੱਖ ਰਖਾਅ ਦੇ ਪ੍ਰੋਸੈਸ ਨੂੰ ਆਸਾਨ ਬਣਾਉਂਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਜਿੱਥੇ ਹਰ ਚੀਜ਼ ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਟਰੂਲੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ (Truly Object-Oriented Programming Language) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

Simula ਨੂੰ ਪਹਿਲੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ Smalltalk ਨੂੰ ਪਹਿਲੀ ਟਰੂਲੀ

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। Java, C++, C#, Phython, R, PHP, Visual Basic. NET, JavaScript, Ruby, Perl, Object Pascal, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB ਅਤੇ Smalltalk ਆਦਿ ਮਹੱਤਵਪੂਰਨ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ।

4.2 ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਸਿਧਾਂਤ (PRINCIPLES OF OBJECT-ORIENTED PROGRAMMING)

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸਭ ਤੋਂ ਵੱਧ ਵਰਤਿਆ ਜਾਣ ਵਾਲਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ ਹੈ। ਇਹ ਕਲਾਸਾਂ ਅਤੇ ਆਬਜੈਕਟਸ ਦੇ ਰੂਪ ਵਿੱਚ ਮੋਡਿਊਲਸ (modules) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਵੱਡੇ ਪੱਧਰ (large scale) ਦੇ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰਾਂ ਨੂੰ ਵਿਕਸਤ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ। ਇਹਨਾਂ ਮੋਡੀਊਲਸ ਦੀ ਮਦਦ ਨਾਲ ਕਈ ਡਿਵੈਲਪਰ ਆਪਸ ਵਿਚ ਮਿਲ ਕੇ ਕੰਮ ਕਰਦੇ ਹੋਏ ਪੂਰਾ ਸਿਸਟਮ ਆਸਾਨੀ ਨਾਲ ਬਣਾ ਸਕਦੇ ਹਨ। ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਐਪਲੀਕੇਸ਼ਨ ਦਾ ਐਂਟੀਟੀਜ਼ ਜਾਂ ਆਬਜੈਕਟਸ ਦੇ ਰੂਪ ਵਿੱਚ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਡਿਜ਼ਾਈਨ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਅਰਥ ਇਹ ਹੈ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਵਿਚ ਐਂਟੀਟੀਜ਼ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਲਾਗੂ ਕਰਨਾ ਹੁੰਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਉਹ ਅਸਲ ਜੀਵਨ ਵਿੱਚ ਵੇਖੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਇਸ ਲਾਗੂਕਰਨ ਵਿਚ ਹਰੇਕ ਐਂਟੀਟੀ ਨੂੰ ਕਿਰਿਆਵਾਂ (actions) ਅਤੇ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (attributes) ਨਾਲ ਜੋੜ ਕੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

OOP ਵਿੱਚ ਕੋਡ ਅਤੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਇਕਾਈ- ਆਬਜੈਕਟ (Object) ਵਿਚ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਆਬਜੈਕਟ-ਅਧਾਰਿਤ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਉਸ ਵਿਚ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਫੰਕਸ਼ਨਾਂ (functions) ਜਾਂ ਪ੍ਰੋਸੀਜ਼ਰਾਂ (procedures) ਵਿੱਚ ਵੰਡਣ ਬਾਰੇ ਨਹੀਂ ਸੋਚਦੇ, ਸਗੋਂ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਉਸਨੂੰ ਆਬਜੈਕਟਸ (objects) ਵਿੱਚ ਵੰਡ ਕੇ ਹੱਲ ਕਰਨ ਬਾਰੇ ਸੋਚਦੇ ਹਾਂ। ਹਰ ਉਹ ਕੰਮ ਜੋ ਅਸੀਂ ਹੱਲ ਕਰਨ ਬਾਰੇ ਸੋਚਦੇ ਹਾਂ ਉਸਨੂੰ ਕਲਾਸਾਂ (classes) ਅਤੇ ਆਬਜੈਕਟਸ (Objects) ਦੇ ਰੂਪ ਵਿੱਚ ਹੱਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।OOP ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਅਸੀਂ ਫੰਕਸ਼ਨੈਲਿਟੀ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਵਾਲੀਆਂ ਕਲਾਸਾਂ (classes) ਬਣਾਉਂਦੇ ਹਾਂ ਅਤੇ ਫਿਰ ਉਸ ਕਲਾਸ ਦਾ ਇੱਕ ਆਬਜੈਕਟ ਬਣਾ ਕੇ ਉਸ ਕਲਾਸ ਦੇ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰ ਸਕਦੇ ਹਾਂ।

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਦੇ ਚਾਰ ਮੁੱਖ ਸਿਧਾਂਤ ਹੁੰਦੇ ਹਨ:

1.	ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction)	2.	ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation)
3.	ਇਹੈਰੀਟੈਂਸ (Inheritance)	4.	ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphim)

ਇਹਨਾਂ ਚਾਰਾਂ ਸਿਧਾਂਤਾਂ ਨੂੰ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਚਾਰ ਥੰਮ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕ ਸਫਲ ਪ੍ਰੋਗਰਾਮਰ ਬਣਨ ਲਈ ਇਹਨਾਂ ਨੂੰ ਸਮਝਣਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੈ। ਪਰ ਇਹਨਾਂ ਸੰਕਲਪਾਂ ਦੀ ਚਰਚਾ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ, ਇਹ ਜਾਣਨਾ ਜ਼ਰੂਰੀ ਹੈ ਕਿ ਕਲਾਸਾਂ (classes) ਅਤੇ ਆਬਜੈਕਟਸ (objects) ਕੀ ਹੁੰਦੇ ਹਨ ? ਆਉ OOP ਦੀ ਬਿਹਤਰ ਸਮਝ ਲਈ ਇਹਨਾਂ ਧਾਰਨਾਵਾਂ ਸੰਬੰਧੀ ਚਰਚਾ ਸ਼ੁਰੂ ਕਰਦੇ ਹਾਂ:

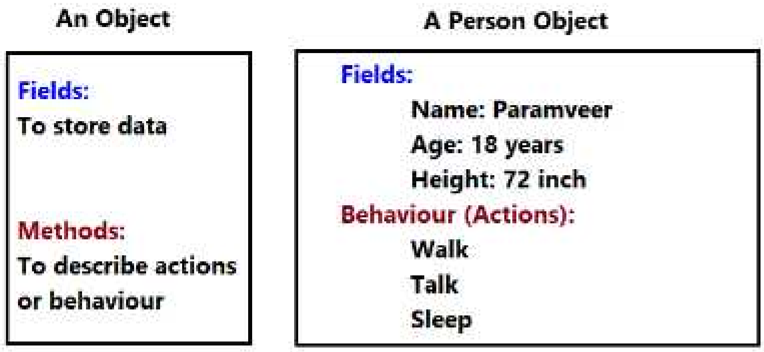
4.2.1 ਆਬਜੈਕਟਸ (Objects)

ਕੋਈ ਵੀ ਅਸਲ ਸੰਸਾਰ ਦੀ ਐਂਟੀਟੀ (entity) ਜਿਸਦੀ ਇਕ ਸਥਿਤੀ/ਸਟੇਟ (state) ਹੁੰਦੀ ਹੈ ਅਤੇ ਜੋ ਕੋਈ ਵਿਵਹਾਰ/ਕਿਰਿਆ (behaviour) ਕਰਦੀ ਹੈ, ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਇੱਕ ਕੁਰਸੀ (chair), ਪੈੱਨ (pen), ਮੇਜ਼ (table), ਕੀਬੋਰਡ (keyboard), ਬਾਈਕ (bike), ਆਦਿ। ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ:

- ❖ ਅਸੀਂ ਆਬਜੈਕਟ ਦੀ ਸਥਿਤੀ/ਸਟੇਟ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਉਸ ਆਬਜੈਕਟ ਦੇ ਡਾਟਾ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਕਿਸੇ ਆਬਜੈਕਟ ਦਾ ਇਹ ਡਾਟਾ ਜਾਂ ਸਟੇਟ ਨੂੰ ਫੀਲਡਸ/ਵੇਰੀਏਬਲਜ਼ **(fields/variables)** ਦੇ ਰੂਪ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਫੀਲਡਸ **(fields)** ਨੂੰ ਐਟਰੀਬਿਊਟਸ (attributes) ਜਾਂ ਪ੍ਰਾਪਰਟੀਜ਼ (*properties*) ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
- ❖ ਆਬਜੈਕਟ ਦੇ ਵਿਵਹਾਰਕਿਰਿਆ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ, ਅਸੀਂ ਕੋਡ (Code) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਹ ਕੋਡ **(Code)** ਮੈਥਡਜ਼ ਜਾਂ ਫੰਕਸ਼ਨਜ਼ **(methods or functions)** ਦੇ ਰੂਪ ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡਜ਼ (methods) ਆਬਜੈਕਟ ਦੇ ਡਾਟਾ ਫੀਲਡਾਂ ਨੂੰ ਅੱਸੈੱਸ (access) ਅਤੇ ਉਹਨਾਂ ਵਿਚ ਸ਼ੋਧ (modify) ਕਰ ਸਕਦੇ ਹਨ।

ਹਰੇਕ ਆਬਜੈਕਟ ਦੀਆਂ ਆਪਣੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (properties) ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਇਹ ਦੱਸਦੀਆਂ ਹਨ ਕਿ ਇਹ ਕੀ ਹੈ ਜਾਂ ਇਹ ਕੀ ਕਰਦਾ ਹੈ ? ਰਨਟਾਈਮ 'ਤੇ ਆਬਜੈਕਟਸ ਬਣਾਉਣ ਲਈ ਟੈਂਪਲੇਟਸ (templates) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਇਹਨਾਂ ਟੈਂਪਲੇਟਸ ਨੂੰ

ਕਲਾਸਾਂ (classes) ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕ ਆਬਜੈਕਟ ਮੈਮੋਰੀ ਵਿੱਚ ਕੁਝ ਥਾਂ ਲੈਂਦਾ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਚਿੱਤਰ ਐਂਟੀਟੀ (entity) ਦੀ ਅਸਲ ਜੀਵਨ (real-life) ਆਧਾਰਿਤ ਉਦਾਹਰਨ ਸਹਿਤ ਇੱਕ ਆਬਜੈਕਟ ਦੀ ਧਾਰਨਾ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ:

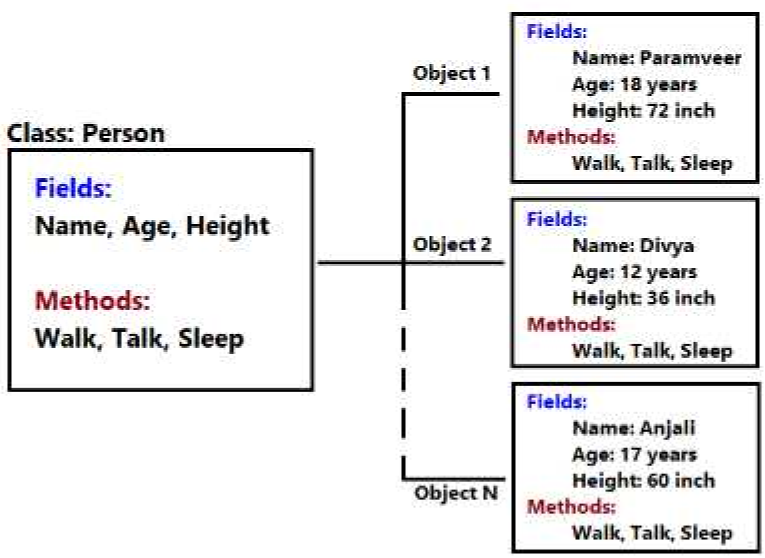


ਚਿੱਤਰ : 4.1 ਆਬਜੈਕਟ ਅਤੇ ਇਸਦੀਆਂ ਉਦਾਹਰਣਾਂ (Person ਆਬਜੈਕਟ)

4.2.2 ਕਲਾਸ (Class): ਇੱਕ ਕਲਾਸ ਉਹਨਾਂ ਆਬਜੈਕਟਸ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦੀ ਹੈ, ਜਿਹਨਾਂ ਵਿੱਚ ਸਮਾਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (same properties) ਅਤੇ ਸਾਂਝਾ ਵਿਵਹਾਰ (common behaviour) ਹੁੰਦਾ ਹੈ। ਕਲਾਸ ਨੂੰ ਇੱਕ ਬਲੂਪ੍ਰਿੰਟ (blueprint) ਵੀ ਮੰਨਿਆ ਜਾ ਸਕਦਾ ਹੈ ਜਿਸ ਤੋਂ ਅਸੀਂ ਇੱਕ ਆਬਜੈਕਟ (objects) ਬਣਾ ਸਕਦੇ ਹਾਂ। ਜਦੋਂ ਇਕ ਕਲਾਸ ਤੋਂ ਆਬਜੈਕਟਸ ਨੂੰ ਬਣਾਈਆਂ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਉਹ ਕਲਾਸ ਤੋਂ ਸਾਰੇ ਵੇਰੀਏਬਲਜ਼ ਅਤੇ ਮੈਥਡਜ਼ (methods) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰ ਲੈਂਦੇ ਹਨ।

ਇਕ ਕਲਾਸ ਉਸਦੇ ਆਬਜੈਕਟਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦੀ ਹੈ ਹਾਲਾਂਕਿ ਇਕ ਆਬਜੈਕਟ ਬਣਾਏ ਜਾਣ ਤੋਂ ਬਾਅਦ ਹੀ ਉਸਦੇ ਮੁੱਲ ਨਿਰਧਾਰਤ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਹਰੇਕ ਆਬਜੈਕਟ ਨੂੰ ਉਸਦੀ ਕਲਾਸ ਦਾ ਇੱਕ ਉਦਾਹਰਣ (instance) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। Person ਕਲਾਸ ਵਿੱਚ ਇਕ ਵਿਸ਼ੇਸ਼ ਵਿਅਕਤੀ ਜਿਵੇਂ ਕਿ: ਪਰਮਵੀਰ, ਦਿਵਿਆ, ਅੰਜਲੀ ਆਦਿ Person ਕਲਾਸ ਦੇ ਆਬਜੈਕਟਸ ਹੋਣਗੇ।

ਇੱਕ ਕਲਾਸ ਕੋਈ ਮੈਮੋਰੀ ਸਪੇਸ ਨਹੀਂ ਵਰਤਦੀ। ਇਹ ਡਾਟਾ ਦੀ ਸਿਰਫ ਇੱਕ ਲੌਜੀਕਲ ਪ੍ਰਤੀਨਿਧਤਾ (logical representation) ਹੈ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇੱਕ ਕਲਾਸ ਆਬਜੈਕਟ ਲਈ ਇੱਕ ਟੈਂਪਲੇਟ (template) ਹੁੰਦੀ ਹੈ ਅਤੇ ਇੱਕ ਆਬਜੈਕਟ ਕਲਾਸ ਦੀ ਇੱਕ ਉਦਾਹਰਣ (instance) ਹੁੰਦੀ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਚਿੱਤਰ ਕਲਾਸ ਅਤੇ ਇਸਦੇ ਆਬਜੈਕਟਸ ਦੀ ਧਾਰਨਾ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ:



ਚਿੱਤਰ : 4.2 ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟ ਦੀ ਧਾਰਨਾ

4.2.3 ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) : ਇਹ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀਆਂ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਧਾਰਨਾਵਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਇਹ ਸਿਧਾਂਤ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਂਟੀਟੀ ਵਿੱਚ ਇੱਕ ਅਸਲ-ਸੰਸਾਰ ਐਂਟੀਟੀ (real world entity) ਨੂੰ ਕਿਵੇਂ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਇੱਕ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਹੈ ਜਿਸ ਵਿੱਚ ਅਸੀਂ ਯੂਜ਼ਰ ਨੂੰ ਆਬਜੈਕਟ ਨਾਲ ਸੰਬੰਧਿਤ ਵੇਰਵੇ ਦਿਖਾਉਣ ਲਈ ਡਾਟਾ ਦੇ ਇੱਕ ਵੱਡੇ ਸੰਗ੍ਰਹਿ ਵਿਚੋਂ ਲੋੜੀਂਦਾ ਡਾਟਾ ਚੁਣਦੇ ਹਾਂ। ਇਹ ਪ੍ਰੋਗਰਾਮਿੰਗ ਜਟਿਲਤਾਵਾਂ (complexity) ਅਤੇ ਕੋਸ਼ਿਸ਼ਾਂ (efforts) ਨੂੰ ਘਟਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ।

ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਐਬਸਟਰੈਕਸ਼ਨ ਨੂੰ ਆਬਜੈਕਟ ਦੇ ਅੰਦਰੂਨੀ ਲਾਗੂਕਰਨ ਨੂੰ ਲੁਕਾਉਣ (hidding internal implementation) ਅਤੇ ਯੂਜ਼ਰਜ਼ ਨੂੰ ਸਿਰਫ ਲੋੜੀਂਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਆਓ ਇੱਕ ਉਦਾਹਰਣ ਦੀ ਮਦਦ ਨਾਲ ਇਸਨੂੰ ਸਮਝਦੇ ਹਾਂ:

ਐਬਸਟਰੈਕਸ਼ਨ ਦੀ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀ ਉਦਾਹਰਨ: ਮੰਨ ਲਵੋ ਕਿ ਅਸੀਂ ਇੱਕ ਬਾਈਕ (bike) ਚਲਾ ਰਹੇ ਹਾਂ। ਇੱਥੇ ਅਸੀਂ ਸਿਰਫ ਬਾਈਕ ਚਲਾਉਣ ਬਾਰੇ ਸੋਚਾਂਗੇ: ਜਿਵੇਂ ਕਿ ਬਾਈਕ ਨੂੰ ਸਟਾਰਟ ਸਟਾਪ ਕਰਨਾ, ਤੇਜ਼ ਕਰਨਾ/ਬ੍ਰੇਕ ਲਗਾਉਣਾ, ਆਦਿ। ਬਾਈਕ ਚਲਾਉਣ ਸਮੇਂ ਅਸੀਂ ਇਹ ਨਹੀਂ ਸੋਚਦੇ ਕਿ ਅਸਲ ਵਿਚ ਇਹ ਸਟਾਰਟ/ਸਟਾਪ ਕਿਵੇਂ ਹੋ ਰਹੀ ਹੈ ਜਾਂ ਤੇਜ਼ ਚਲਾਉਣ/ਬ੍ਰੇਕ ਲਗਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਅੰਦਰੂਨੀ ਤੌਰ 'ਤੇ ਕਿਵੇਂ ਕੰਮ ਕਰ ਰਹੀ ਹੈ। ਸਾਨੂੰ ਉਹਨਾਂ ਵੇਰਵਿਆਂ ਵਿੱਚ ਕੋਈ ਦਿਲਚਸਪੀ ਨਹੀਂ ਹੁੰਦੀ, ਅਸੀਂ ਸਿਰਫ ਬਾਈਕ ਦੀ ਵਰਤੋਂ ਕਰਨ ਬਾਰੇ ਹੀ ਸੋਚਦੇ ਹਾਂ।

4.2.4 ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation) : ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਸਮੇਟਣ (wrapping) ਦੀ ਇੱਕ ਪ੍ਰਕਿਰਿਆ ਹੈ। ਇਸ ਸਿੰਗਲ ਯੂਨਿਟ ਨੂੰ ਕਲਾਸ (class) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਇਸਨੂੰ ਇੱਕ ਅਜਿਹੇ ਸੁਰੱਖਿਆ ਰੈਪਰ (protective wrapper) ਵਜੋਂ ਮੰਨ ਸਕਦੇ ਹਾਂ ਜੋ ਰੈਪਰ ਦੇ ਅੰਦਰ ਪਰਿਭਾਸ਼ਿਤ ਕੋਡ ਦੀ ਰੈਂਡਮ ਅਸੈਸ ਨੂੰ ਬਾਹਰੋਂ ਰੋਕਦਾ ਹੈ। ਇਹ ਇੱਕ ਕੈਪਸੂਲ ਵਾਂਗ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਕਈ ਤਰ੍ਹਾਂ ਦੀਆਂ ਦਵਾਈਆਂ ਦਾ ਮਿਸ਼ਰਣ ਹੁੰਦਾ ਹੈ।

ਐਨਕੈਪਸੁਲੇਸ਼ਨ ਡਾਟਾ ਦੀ ਸੁਰੱਖਿਆ ਨੂੰ ਵਧਾਉਂਦੀ ਹੈ, ਕਿਉਂਕਿ ਇਸਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਸਿੰਗਲ ਟਾਸਕ ਨਾਲ ਸਬੰਧਤ ਹਰ ਚੀਜ਼ ਨੂੰ ਇਕੱਠਾ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਲੋੜ ਅਨੁਸਾਰ ਡਾਟਾ ਦਾ ਅਸੈਸ ਪ੍ਰਦਾਨ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਅਤੇ ਇਸ ਕੰਮ ਲਈ ਡਾਟਾ ਛੁਪਾਉਣ (Data Hiding) ਦੀ ਧਾਰਨਾ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦਾ ਹੈ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਬਿਆਨ ਕੀਤਾ ਜਾ ਸਕਦਾ

Encapsulation = Data Hiding + Abstraction

ਇਸ ਤਰ੍ਹਾਂ ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਸਾਡੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਛੁਪਾ ਕੇ (selective hiding of properties and methods) ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਲਪੇਟ (wrapping) ਕੇ ਰੱਖਦੀ ਹੈ, ਜਿਸਨੂੰ ਕਲਾਸ (class) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਅਸੈੱਸ ਮੋਡੀਫਾਇਰ **private** ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਾਟਾ ਨੂੰ ਛੁਪਾ ਕੇ ਰੱਖਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਪ੍ਰਾਈਵੇਟ (Private) ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਕਲਾਸ ਤੋਂ ਬਾਹਰ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਦੁਆਰਾ ਐਕਸੈਸ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਆਓ ਇੱਕ ਉਦਾਹਰਣ ਦੀ ਮਦਦ ਨਾਲ ਸਮਝੀਏ:

ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦੀ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀ ਉਦਾਹਰਨ : ਬਾਜ਼ਾਰ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਸਿਹਤ ਸਮੱਸਿਆਵਾਂ ਦੇ ਇਲਾਜ ਲਈ ਕੈਪਸੂਲਜ਼ (Capsules) ਉਪਲਬਧ ਹਨ। ਕੈਪਸੂਲ ਵਿੱਚ ਇੱਕ ਪੂਰੀ ਦਵਾਈ ਬਣਾਉਣ ਲਈ ਵੱਖ-ਵੱਖ ਰਚਨਾਵਾਂ ਦਾ ਸਮੂਹ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਕਿਸੇ ਖਾਸ ਸਿਹਤ ਸਮੱਸਿਆ ਨੂੰ ਠੀਕ ਕਰਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਕੈਪਸੂਲ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਰਚਨਾਵਾਂ ਦੇ ਸਮੂਹ ਨੂੰ ਇੱਕਠਾ ਕਰਕੇ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਦੇ ਤੌਰ ਤੇ ਰੱਖਣਾ ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦਾ ਹੀ ਇੱਕ ਰੂਪ ਹੈ।

ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ:

1. ਇਹ ਡਾਟਾ ਲੁਕਾਉਣ ਦੀ ਧਾਰਨਾ (concept of Data Hiding) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਕਲਾਸ ਦੇ ਮੈਂਬਰਾਂ ਨੂੰ ਵਾਧੂ ਸੁਰੱਖਿਆ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।
2. ਇਹ ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਇੱਕਠਾ ਕਰਕੇ ਰੱਖਦਾ ਹੈ।
3. ਇਹ ਡਾਟਾ ਨੂੰ ਵਾਧੂ ਸੁਰੱਖਿਆ ਮੁਹਈਆ ਕਰਵਾਉਂਦਾ ਹੈ ਜੋ ਅਣਅਧਿਕਾਰਤ (unauthorized) ਲੋਕਾਂ ਦੁਆਰਾ ਡਾਟਾ ਨੂੰ ਅਸੈਸ ਕਰਨ 'ਤੇ ਪਾਬੰਦੀ (Restrictions) ਲਗਾਉਂਦਾ ਹੈ।

4.2.5 ਇਨਹੇਰੀਟੈਂਸ (Inheritance): ਇਹੇਰੀਟੈਂਸ ਇੱਕ ਅਜਿਹੀ ਵਿਧੀ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੇ ਪੇਰੈਂਟ (parent) ਆਬਜੈਕਟ ਦੀਆਂ ਸਾਰੀਆਂ ਅਵਸਥਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰਾਂ (states and behaviours) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਇੱਕ ਬੱਚੇ ਨੂੰ ਉਸਦੇ ਮਾਤਾ-ਪਿਤਾ ਦੇ ਗੁਣ ਵਿਰਾਸਤ (inherits) ਵਿੱਚ ਮਿਲਦੇ ਹਨ।

ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਇਨਹੇਰੀਟੈਂਸ ਇਕ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਹੈ ਜਿਸ ਵਿਚ ਮੌਜੂਦਾ ਕਲਾਸਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ

ਨਵੀਆਂ ਕਲਾਸਾਂ ਬਣਾਈਆਂ (creating new classes using the existing classes) ਜਾਂਦੀਆਂ ਹਨ। ਇਸ ਪ੍ਰਕਿਰਿਆ ਦੀ ਵਰਤੋਂ ਨਾਲ ਨਵੀਂ ਕਲਾਸ ਮੌਜੂਦਾ ਕਲਾਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਗ੍ਰਹਿਣ ਕਰਦੀ ਹੈ। ਇਹੈਰੀਟੈਂਸ ਨਾਲ ਅਸੀਂ ਮੌਜੂਦਾ ਕਲਾਸ ਦੇ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ (fields and methods) ਦੀ ਮੁੜ ਵਰਤੋਂ (reuse) ਕਰ ਸਕਦੇ ਹਾਂ। ਇਸ ਤਰ੍ਹਾਂ ਇਹੈਰੀਟੈਂਸ ਮੁੜ-ਵਰਤੋਂਯੋਗਤਾ (Reusability) ਦੀ ਸਹੂਲਤ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ ਅਤੇ ਇਹ OOP ਦੀ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਧਾਰਨਾ ਮੰਨੀ ਜਾਂਦੀ ਹੈ।

ਮੰਨ ਲਓ ਇੱਕ ਪੇਰੈਂਟ ਕਲਾਸ (parent class) ਉਪਲਬਧ ਹੈ ਜਿਸ ਵਿਚ ਕੁੱਝ ਮੈਥਡਜ਼ (methods) ਉਪਲਬਧ ਹਨ ਅਤੇ ਹੁਣ ਅਸੀਂ ਇੱਕ ਨਵੀਂ ਕਲਾਸ ਬਣਾਵਾਂਗੇ, ਜਿਸਨੂੰ ਚਾਈਲਡ ਕਲਾਸ (child class) ਕਹਾਂਗੇ, ਜਿਸ ਵਿਚ ਇਸ ਕਲਾਸ ਦੇ ਆਪਣੇ ਮੈਥਡਜ਼ (methods) ਹੋਣਗੇ। ਹੁਣ ਜਦੋਂ ਇੱਕ ਚਾਈਲਡ ਕਲਾਸ ਨੂੰ ਪੇਰੈਂਟ ਕਲਾਸ ਤੋਂ ਇਨਹੈਰਿਟ (inherit) ਕੀਤਾ ਜਾਵੇਗਾ, ਤਾਂ ਪੇਰੈਂਟ ਕਲਾਸ ਨਾਲ ਸਬੰਧਤ ਨਾਨ ਪ੍ਰਾਇਵੇਟ ਮੈਥਡਜ਼ ਚਾਈਲਡ ਕਲਾਸ ਵਿੱਚ ਚਾਈਲਡ ਕਲਾਸ ਦੇ ਆਪਣੇ ਮੈਥਡਜ਼ ਨਾਲ ਉਪਲਬਧ ਹੋਣਗੇ। ਮੌਜੂਦਾ ਕਲਾਸਾਂ (existing classes) ਤੋਂ ਨਵੀਆਂ ਵਧੀਆਂ ਕਲਾਸਾਂ (new enhanced classes) ਬਣਾਉਣ ਦੀ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਇਨਹੈਰੀਟੈਂਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਇਨਹੈਰੀਟੈਂਸ ਲਈ ਦੋ ਸ਼ਬਦਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ – ਬੇਸ ਕਲਾਸ (Base Class) ਅਤੇ ਡਰਾਈਵਡ ਕਲਾਸ (Derived Class)।

- **ਬੇਸ ਕਲਾਸ (Base Class)** - ਇਸਨੂੰ ਪੇਰੈਂਟ ਕਲਾਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੁੱਖ ਕਲਾਸ ਹੁੰਦੀ ਹੈ।ਜਿਸ ਵਿਚ ਬੁਨਿਆਦੀ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ (basic properties and methods) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੁੰਦਾ ਹੈ।
- **ਡਰਾਈਵਡ ਕਲਾਸ (Derived Class)** - ਇਹ ਬੇਸ ਕਲਾਸ ਦੀ ਐਕਸਟੈਨਸ਼ਨ ਹੁੰਦੀ ਹੈ ਅਤੇ ਜਿਸਨੂੰ ਚਾਈਲਡ ਕਲਾਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਵਿੱਚ ਆਪਣੀਆਂ ਖੁੱਦ ਦੀਆਂ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਦੇ ਨਾਲ ਉਹ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਵੀ ਹੁੰਦੇ ਹਨ ਜੋ ਬੇਸ ਕਲਾਸ ਵਿੱਚ ਮੌਜੂਦ ਹੁੰਦੇ ਹਨ।

ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀ ਉਦਾਹਰਨ: ਅਸੀਂ ਸਭ ਨੇ ਮੋਬਾਈਲ ਫ਼ੋਨਾਂ ਲਈ ਅੱਪਡੇਟ ਦੇਖੇ ਹਨ। ਸ਼ੁਰੂ ਵਿੱਚ ਮੋਬਾਈਲ ਫ਼ੋਨ ਸਿਰਫ ਗੱਲ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਸਨ ਅਤੇ ਫਿਰ ਮੀਡੀਆ ਅਸੈਸ, ਇੰਟਰਨੈਟ, ਕੈਮਰਾ ਆਦਿ ਲਈ ਵਰਤੇ ਜਾਣ ਲੱਗੇ। ਇਹ ਉਹ ਵਿਕਾਸ ਹੈ ਜਿਸ ਵਿਚ ਪੂਰੀ ਕਾਰਜਕੁਸ਼ਲਤਾ ਦਾ ਮੁੜ ਨਿਰਮਾਣ ਕੀਤੇ ਬਿਨਾਂ, ਮੌਜੂਦਾ ਤਕਨੀਕ ਵਿਚ ਕੁੱਝ ਵਾਧੂ ਵਿਸ਼ੇਸ਼ਤਾ ਜੋੜ ਕੇ ਨਵਾਂ ਉਤਪਾਦ ਬਣਾ ਕੇ ਪੇਸ਼ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਲਈ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਹੀ ਇੱਕ ਉਦਾਹਰਣ ਹੈ।

4.2.6 ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism): ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਕਿਸੇ ਆਬਜੈਕਟ ਦੀ ਕਈ ਰੂਪ ਧਾਰਨ ਕਰਨ ਦੀ ਯੋਗਤਾ ਹੁੰਦੀ ਹੈ। ਇਹ ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸਿਧਾਂਤ ਦਾ ਸਭ ਤੋਂ ਜ਼ਰੂਰੀ ਸੰਕਲਪ ਹੈ। ਪੌਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਸ਼ਬਦ ਦੋ ਸ਼ਬਦਾਂ ਤੋਂ ਬਣਿਆ ਹੈ: **Poly + morph** (ਪੌਲੀ + ਮੋਰਫ)। ਇੱਥੇ 'poly' ਪੌਲੀ ਦਾ ਅਰਥ ਹੈ 'ਬਹੁਤ ਸਾਰੇ' ਅਤੇ (ਮੋਰਫ) ਦਾ ਅਰਥ ਹੈ 'ਰੂਪ'। ਇਸ ਤਰ੍ਹਾਂ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਦਾ ਅਰਥ ਹੁੰਦਾ ਹੈ ਕਈ ਰੂਪ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਇੱਕ ਵਸਤੂ ਦੇ ਕਈ ਰੂਪਾਂ ਨੂੰ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਜਾਂ ਮੈਥਡ ਨਾਲ ਇੱਕ ਤੋਂ ਵੱਧ ਨਾਮ ਜੁੜੇ ਹੋ ਸਕਦੇ ਹਨ। ਇਹ ਹੋਰ ਕੁੱਝ ਨਹੀਂ ਸਗੋਂ **ਪੋਲੀਮੋਰਫਿਜ਼ਮ** ਹੀ ਹੁੰਦਾ ਹੈ। ਆਉ ਅਸੀਂ ਇੱਕ ਅਸਲ-ਜੀਵਨ ਦੀ ਉਦਾਹਰਨ ਨਾਲ ਪੋਲੀਮੋਫਿਜ਼ਮ ਦੀ ਧਾਰਨਾ ਨੂੰ ਸਮਝੀਏ:

ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਦੀਆਂ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀਆਂ ਉਦਾਹਰਣਾਂ:

- ਇੱਕ ਅਧਿਆਪਕ ਦਾ ਵਿਦਿਆਰਥੀਆਂ ਨਾਲ ਵਿਵਹਾਰ।
- ਇੱਕ ਅਧਿਆਪਕ ਦਾ ਆਪਣੇ ਸੀਨੀਅਰਾਂ ਨਾਲ ਵਿਵਹਾਰ।

ਇੱਥੇ ਅਧਿਆਪਕ ਇੱਕ ਆਬਜੈਕਟ ਹੈ ਪਰ ਉਸਦਾ ਵਿਵਹਾਰ ਵੱਖ-ਵੱਖ ਸਥਿਤੀਆਂ ਵਿੱਚ ਵੱਖਰਾ-ਵੱਖਰਾ ਹੋਵੇਗਾ, ਭਾਵ ਉਸਦਾ ਵਿਵਹਾਰ ਵਿਦਿਆਰਥੀਆਂ ਨਾਲ ਵੱਖਰਾ ਹੋਵੇਗਾ ਅਤੇ ਆਪਣੇ ਸੀਨੀਅਰਾਂ ਨਾਲ ਵੱਖਰਾ ਹੋਵੇਗਾ, ਜੋ ਕੇ ਅਧਿਆਪਕ ਦੇ ਇਕ ਤੋਂ ਵੱਧ ਰੂਪਾਂ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ।

ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਦੀਆਂ ਕਿਸਮਾਂ (Types of Polymorphism):

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਦੋ ਤਰ੍ਹਾਂ ਦੇ ਹੁੰਦੇ ਹਨ:

1. ਕੰਪਾਈਲ-ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Compile-Time Polymorphism): ਇਸਨੂੰ ਸਟੈਟਿਕ (static) ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਜਾਂ ਅਰਲੀ ਬਾਈਡਿੰਗ (early binding) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਕੰਪਾਈਲਰ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਦੇ ਸਮੇਂ ਮੈਥਡ ਸਿਗਨੇਚਰ (method signature) ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ ਅਤੇ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਦਿੱਤੇ ਮੈਥਡ ਨੂੰ ਕਾਲ (given method call) ਕਰਨ ਲਈ ਕਿਸ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਵੇ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਸ ਕਿਸਮ ਦੇ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਵਿੱਚ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕਰਨ ਸਮੇਂ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨਾਲ ਬੱਝੀਆ (object is bound with its functionality) ਹੁੰਦਾ ਹੈ।

ਕੰਪਾਈਲ ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ **(Method Overloading)** ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਸਿਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading) : ਜਦੋਂ ਇੱਕ ਤੋਂ ਵੱਧ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕੋ ਨਾਮ ਨਾਲ ਘੋਸ਼ਿਤ (declared) ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਪਰ ਉਹਨਾਂ ਨੂੰ ਪੈਰਾਮੀਟਰਾਂ (Parameters) ਦੀ ਇੱਕ ਵੱਖਰੀ ਸੰਖਿਆ ਜਾਂ ਪੈਰਾਮੀਟਰ ਦੇ ਵੱਖਰੇ ਡਾਟਾ-ਟਾਈਪ ਦੇ ਨਾਲ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਉਸਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

2. ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Run-Time Polymorphism) : ਇਸਨੂੰ ਡਾਇਨਾਮਿਕ (dynamic) ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਜਾਂ ਲੇਟ ਬਾਈਡਿੰਗ (Late Binding) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਰਨਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਵਿੱਚ, ਰਨਟਾਈਮ ਦੌਰਾਨ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕਰਨ ਲਈ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਸ ਕਿਸਮ ਦੇ ਪੋਲੀਮੋਫਿਜ਼ਮ ਵਿੱਚ ਰਨ ਟਾਈਮ ਸਮੇਂ ਇੱਕ ਆਬਜੈਕਟ ਨੂੰ ਉਸਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨਾਲ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ।ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਫਿਜ਼ਮ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Method Overriding) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Method Overloading) : ਜਦੋਂ ਇੱਕ ਤੋਂ ਵੱਧ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕੋ ਨਾਮ ਅਤੇ ਇੱਕੋ ਹਸਤਾਖਰ (same signature) ਨਾਲ ਵੱਖਰੀਆਂ ਕਲਾਸਾਂ ਵਿੱਚ ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸ ਕੀਰਿਆ ਨੂੰ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਡਰਾਈਵਡ ਕਲਾਸ ਵਿਚ ਇਕੋ ਨਾਮ ਨਾਲ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਮੈਥਡ ਬੇਸ ਕਲਾਸ ਵਿਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤੇ ਗਏ ਮੈਥਡ ਨੂੰ ਓਵਰਰਾਈਡ ਕਰੇਗਾ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੋਇਆ ਕਿ ਬੇਸ ਕਲਾਸ ਦੇ ਮੈਥਡ ਨੂੰ ਡਰਾਈਵਡ ਕਲਾਸ ਦੇ ਮੈਥਡ ਨਾਲ ਸ਼ੈਡੋਡ (shadowed) ਕੀਤਾ ਗਿਆ ਹੈ ਅਤੇ ਜਦੋਂ ਇਸ ਓਵਰਰੀਡਨ ਮੈਥਡ (Overidden Method) ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਵੇਗਾ ਤਾਂ ਉਸਨੂੰ ਓਵਰਰਾਈਡ ਮੈਥਡ ਦੇ ਅਧਾਰ ਤੇ ਕਾਲ ਕੀਤਾ ਜਾਵੇਗਾ।

4.3 ਜਾਵਾ ਨਾਲ ਜਾਣ ਪਛਾਣ (INTRODUCTION TO JAVA)

JAVA ਨੂੰ ਸਾਲ 1995 ਵਿੱਚ ਸਨ ਮਾਈਕੋਸਿਸਟਮਜ਼ **(Sun Microsystems)** ਕੰਪਨੀ ਵਿੱਚ ਜੇਮਸ ਗੋਸਲਿੰਗ (James Gosling) ਦੁਆਰਾ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਸੀ। ਬਾਅਦ ਵਿੱਚ ਇਸਦੀ ਮਲਕੀਅਤ ਓਰੇਕਲ (Oracle) ਕਾਰਪੋਰੇਸ਼ਨ ਦੁਆਰਾ ਹਾਸਿਲ ਕਰ ਲਈ ਗਈ ਸੀ। ਜਾਵਾ ਇੱਕ ਸਧਾਰਨ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ। ਜਾਵਾ ਇੱਕ ਆਮ ਉਦੇਸ਼ (general-purpose), ਕਲਾਸ-ਅਧਾਰਿਤ (class-based), ਆਬਜੈਕਟ-ਅਧਾਰਿਤ (object-oriented) ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ, ਜੋ ਕਿ ਵਿੰਡੋਜ਼, ਮੈਕ (Mac) ਅਤੇ ਲਾਇਨੇਕਸ (Linux) ਵਰਗੇ ਵੱਖ ਵੱਖ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮਾਂ 'ਤੇ ਕੰਮ ਕਰਦੀ ਹੈ। JAVA ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ, ਕੰਪਾਇਲੇਸ਼ਨ, ਅਤੇ ਡੀਬੱਗਿੰਗ ਕਰਨਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ। ਇਹ ਮੁੜ ਵਰਤੋਂ ਯੋਗ ਕੋਡ (reusable code) ਅਤੇ ਮਾਡਿਊਲਰ (modular) ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ।JAVA ਵਿਚ ਇਕ ਵਾਰ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਤੋਂ ਬਾਅਦ ਕਿਸੇ ਵੀ ਤਰ੍ਹਾਂ ਦੀ ਮਸ਼ੀਨ ਉਪਰ ਚੱਲਣ ਯੋਗ (Write Once Run Anywhere) ਹੁੰਦਾ ਹੈ। ਸਾਧਾਰਣ ਸ਼ਬਦਾਂ ਵਿਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਕੰਪਾਇਲ ਕੀਤਾ ਗਿਆ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਨੂੰ ਸਪੋਰਟ ਕਰਨ ਵਾਲੇ ਸਾਰੇ ਪਲੇਟਫਾਰਮਾਂ 'ਤੇ ਚੱਲ ਸਕਦਾ ਹੈ। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਈਲ ਹੋਣ ਤੋਂ ਬਾਅਦ ਬਾਈਟ ਕੋਡ ਵਿਚ ਤਬਦੀਲ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿਸੇ ਵੀ ਜਾਵਾ ਵਰਚੁਅਲ ਮਸ਼ੀਨ (Java Virtual Machine) 'ਤੇ ਚੱਲ ਸਕਦਾ ਹੈ। ਜਾਵਾ ਦਾ ਸਿੰਟੈਕਸ C/C++ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੇ ਸਮਾਨ ਹੁੰਦਾ ਹੈ।

ਅਸੀਂ ਜਾਵਾ ਦੀ ਵਰਤੋਂ ਹੇਠਾਂ ਦਿਤੀਆਂ ਕਿਸਮਾਂ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਡਿਵੈਲਪ ਕਰਨ ਲਈ ਕਰ ਸਕਦੇ ਹਾਂ:

- ❖ ਡੈਸਕਟਾਪ ਐਪਲੀਕੇਸ਼ਨਾਂ (Desktop Applications)
- ❖ ਵੈੱਬ ਐਪਲੀਕੇਸ਼ਨਾਂ (Web Applications)
- ❖ ਮੋਬਾਈਲ ਐਪਲੀਕੇਸ਼ਨਾਂ (Andriod Mobile Application)
- ❖ ਵੈੱਬ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਸਰਵਰ (Web and Applications Server)
- ❖ ਬਿੱਗ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਐਪਲੀਕੇਸ਼ਨਾਂ (Big data Proceesing Application)
- ❖ ਏਮਬੈਡਡ (Embedded) ਸਿਸਟਮ ਅਤੇ ਹੋਰ ਵੀ ਬਹੁਤ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਜਾਵਾ ਵਿਚ ਤਿਆਰ ਕੀਤੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ।

4.3.1 ਜਾਵਾ ਵਾਤਾਵਰਨ (Java Environment)

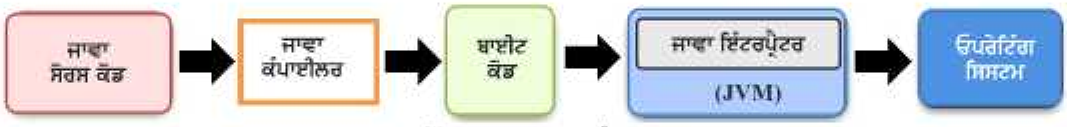
C ਅਤੇ C++ ਸਮੇਤ ਕਈ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੇ ਵਿਪਰੀਤ, ਜਦੋਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਹ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਪਲੇਟਫਾਰਮ-ਮਸ਼ੀਨ (platform specific machine) ਵਿੱਚ ਕੰਪਾਇਲ ਨਹੀਂ ਹੁੰਦਾ; ਇਸ ਦੀ ਬਜਾਏ ਇਸ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਬਾਈਟ-ਕੋਡ (platform independent byte-code) ਵਿੱਚ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਬਾਈਟ-ਕੋਡ ਨੂੰ ਫਿਰ ਜਾਵਾ ਵਰਚੁਅਲ ਮਸ਼ੀਨ (JVM) ਦੁਆਰਾ ਇੰਟਰਪ੍ਰੇਟ (ਐਗਜ਼ੀਕਿਊਟ) ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਬਾਈਟ-ਕੋਡ ਕਿਸੇ ਵੀ ਪਲੇਟਫਾਰਮ 'ਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਭਾਵੇਂ ਉਹ ਵਿੰਡੋਜ਼ (Windows), ਲੀਨਕਸ (Linux) ਜਾਂ ਮੈਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ (macOS)

ਹੋਵੇ।ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜੇਕਰ ਅਸੀਂ ਵਿੰਡੋਜ਼ ਉੱਤੇ ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਇਸਨੂੰ ਬਿਨ੍ਹਾਂ ਕੋਈ ਬਦਲਾਵ ਕੀਤੇ ਲੀਨਕਸ ਉੱਤੇ ਵੀ ਚਲਾ ਸਕਦੇ ਹਾਂ ਅਤੇ ਠੀਕ ਇਸੇ ਤਰ੍ਹਾਂ ਲੀਕਨਸ ਪਲੇਟਫਾਰਮ ਉਪਰ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਈਲ ਕਰਕੇ ਉਸ ਕੋਡ ਨੂੰ ਵਿੰਡੋਜ਼ ਪਲੇਟਫਾਰਮ ਉਪਰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਹਰੇਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦਾ ਇੱਕ ਵੱਖਰਾ JVM ਹੁੰਦਾ ਹੈ, ਪਰ ਬਾਈਟ-ਕੋਡ ਦੇ ਲਾਗੂ (exection) ਹੋਣ ਤੋਂ ਬਾਅਦ ਸਾਰੇ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੁਆਰਾ ਪੈਦਾ ਕੀਤੀ ਗਈ ਆਉਟਪੁੱਟ ਇੱਕੋ ਜਿਹੀ ਹੁੰਦੀ ਹੈ। ਇਸ ਲਈ ਜਾਵਾ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਭਾਸ਼ਾ (platform indepent language) ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ।

ਲੀਨਕਸ ਅਤੇ ਵਿੰਡੋਜ਼ ਪਲੇਟਫਾਰਮਾਂ ਵਿਚ ਜਾਵਾ ਦੀਆਂ ਵਾਤਾਵਰਣ ਸੈਟਿੰਗਾਂ ਨੂੰ ਹੇਠਾਂ ਦਿਖਾਇਆ ਗਿਆ ਹੈ।JVM, JRE ਅਤੇ JDK ਇਹ ਸਾਰੇ JAVA ਦੇ ਜ਼ਰੂਰੀ ਭਾਗ ਹਨ ਅਤੇ ਇਹ ਸਾਰੇ ਪਲੇਟਫਾਰਮ-ਨਿਰਭਰ (platform-dependent) ਹਨ ਕਿਉਂਕਿ ਹਰੇਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੀ ਸੰਰਚਨਾ (configuration) ਵੱਖਰੀ ਹੁੰਦੀ ਹੈ। ਪਰ ਜਾਵਾ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ (platform indepent) ਹੈ। ਇਸ ਪਾਠ ਵਿਚ ਅੱਗੇ ਵਧਣ ਤੋਂ ਪਹਿਲਾਂ ਸਾਨੂੰ ਜਾਵਾ ਵਾਤਾਵਰਣ ਨਾਲ ਸੰਬੰਧਤ ਕੁੱਝ ਗੱਲਾਂ ਸਪੱਸ਼ਟ ਹੋਣੀਆਂ ਚਾਹੀਦੀਆਂ ਹਨ ਜੋ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਦੀ ਵਰਤੋਂ ਨਾਲ ਬਿਹਤਰ ਸਮਝੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ:

ਚਿੱਤਰ : 4.3 ਜਾਵਾ ਦੇ ਵਾਤਾਵਰਣ ਦੀ ਸਟਰਕਚਰ (JDJ, JRE ਅਤੇ JVM)

- **JDK (ਜਾਵਾ ਡਿਵੈਲਪਮੈਂਟ ਕਿੱਟ) :** ਇਹ JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਮੁੱਖ ਪਲੇਟਫਾਰਮ ਕੰਪੋਨੈਂਟ ਹੈ। ਇਸਦਾ ਮੁੱਖ ਭਾਗ ਜਾਵਾ ਕੰਪਾਈਲਰ ਹੈ। ਇਸ ਵਿੱਚ JRE + ਡਿਵੈਲਪਮੈਂਟ ਟੂਲਜ਼ ਸ਼ਾਮਲ ਹਨ। ਡਿਵੈਲਪਮੈਂਟ ਟੂਲਜ਼ ਵਿੱਚ JAVA ਕੰਪਾਈਲਰ (JAVAC), JAVADOC, Debugger ਆਦਿ ਸ਼ਾਮਲ ਹਨ। JDK ਕੰਪੋਨੈਂਟ ਸਾਫਟਵੇਅਰ ਡਿਵੈਲਪਰਾਂ ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ।
- **JRE (ਜਾਵਾ ਰਨਟਾਈਮ ਐਨਵਾਇਰਮੈਂਟ) :** JRE ਵਿੱਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਜਾਵਾ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।JRE ਨੂੰ ਸਿਰਫ਼ JAVA ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਇੱਕ ਸਟੈਂਡਅਲੋਨ ਕੰਪੋਨੈਂਟ (standalone component) ਵਜੋਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ, ਪਰ ਇਹ JDK ਦਾ ਵੀ ਹਿੱਸਾ ਹੁੰਦੀ ਹੈ। JDK ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ JRE ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਕਿਉਂਕਿ JAVA ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣਾ ਪ੍ਰੋਗਰਾਮ ਡਿਵੇਲਪਮੈਂਟ ਦਾ ਹੀ ਹਿੱਸਾ ਹੁੰਦਾ ਹੈ। ਇਹ ਕੰਪੋਨੈਂਟ ਅੰਤਮ ਉਪਭੋਗਤਾਵਾਂ (End-Users) ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ।
- **JVM (ਜਾਵਾ ਵਰਚੁਅਲ ਮਸ਼ੀਨ):** JVM ਇੱਕ ਐਬਸਟਰੈਕਟ (abstract) ਮਸ਼ੀਨ ਹੁੰਦੀ ਹੈ। ਇਹ ਇੱਕ ਸਪੈਸੀਫੀਕੇਸ਼ਨ (specification) ਹੈ ਜੋ ਇਕ ਅਜਿਹਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ (runtime enviroment) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਜਾਵਾ ਬਾਈਟਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।JVM ਬਹੁਤ ਸਾਰੇ ਵੱਖ-ਵੱਖ ਹਾਰਡਵੇਅਰ ਅਤੇ ਸਾਫਟਵੇਅਰ ਪਲੇਟਫਾਰਮਾਂ ਲਈ ਉਪਲਬਧ ਹਨ। ਅਸੀਂ ਇਹ ਵੀ ਕਹਿ ਸਕਦੇ ਹਾਂ JVM ਕੰਪੋਨੈਂਟ JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਇੱਕ ਰਨ-ਟਾਈਮ ਇੰਜਣ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ।JVM ਉਹ ਕੰਪੋਨੈਂਟ ਹੁੰਦਾ ਹੈ ਜੋ ਅਸਲ ਵਿੱਚ ਜਾਵਾ ਕੋਡ ਵਿੱਚ ਮੌਜੂਦ ਮੇਨ ਮੈਥਡ (main method) ਨੂੰ ਕਾਲ ਕਰਦਾ ਹੈ।JVM ਕੰਪੋਨੈਂਟ JRE (ਜਾਵਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ) ਦਾ ਹੀ ਇੱਕ ਹਿੱਸਾ ਹੁੰਦਾ ਹੈ। ਜਾਵਾ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ WORA (Write Once Run Anywhere) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਰ ਇੱਕ ਸਿਸਟਮ ਤੇ ਜਾਵਾ ਕੋਡ ਡਿਵੇਲਪ ਕਰਕੇ ਉਸ ਵਿਚ ਬਿਨਾਂ ਕੋਈ ਬਦਲਾਵ ਕਿਤੇ ਕਿਸੇ ਹੋਰ ਜਾਵਾ-ਸਮਰਥਿਤ (JVM-enabled) ਸਿਸਟਮ ਤੇ ਚੱਲਾ ਸਕਦਾ ਹੈ। ਇਹ ਸਭ JVM ਦੇ ਕਾਰਨ ਹੀ ਸੰਭਵ ਹੋਇਆ ਹੈ।



ਚਿੱਤਰ : 4.4 JVM ਫੰਕਸ਼ਨ

4.4 ਜਾਵਾ ਦਾ ਇਤਿਹਾਸ (HISTORY OF JAVA) :

ਜਾਵਾ ਨੂੰ ਜੂਨ 1991 ਵਿੱਚ ਜੇਮਜ਼ ਗੋਸਲਿੰਗ ਅਤੇ ਉਸਦੀ ਟੀਮ (ਪੈਟਰਿਕ ਨੌਟਨ, ਕ੍ਰਿਸ ਵਾਰਥ, ਮਾਈਕ ਸ਼ੈਰੀਡਨ, ਅਤੇ ਐਡ ਫਰੈਂਕ) ਦੁਆਰਾ "ਓਕ (Oak)" ਨਾਮਕ ਇੱਕ ਪ੍ਰੋਜੈਕਟ ਵਜੋਂ ਸ਼ੁਰੂ ਕੀਤਾ ਗਿਆ ਸੀ। ਇਹ ਪ੍ਰੋਜੈਕਟ ਇੱਕ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਭਾਸ਼ਾ (platfrom-independent language) ਵਿਕਸਿਤ ਕਰਨ ਅਤੇ ਖਪਤਕਾਰ ਇਲੈਕਟ੍ਰਾਨਿਕ ਯੰਤਰਾਂ (Consumer Electronic Devices) ਲਈ ਏਮਬੈਡਡ (embedded) ਸਾਫਟਵੇਅਰ ਬਣਾਉਣ ਦੇ ਵਿਚਾਰ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤਾ ਗਿਆ ਸੀ। ਗੋਸਲਿੰਗ ਦਾ ਟੀਚਾ ਇੱਕ ਵਰਚੁਅਲ ਮਸ਼ੀਨ ਅਤੇ ਇੱਕ ਅਜਿਹੀ ਭਾਸ਼ਾ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਸੀ ਜਿਸ ਵਿੱਚ ਜਾਣੀ-ਪਛਾਣੀ ਭਾਸ਼ਾ C ਵਰਗੀ ਨੋਟੇਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਵੇ ਪਰ ਇਹ C/C ++ ਨਾਲੋਂ ਵੱਧ ਇਕਸਾਰ ਅਤੇ ਸਰਲ ਹੋਵੇ। ਇਸ ਨੂੰ ਵਿਕਸਤ ਕਰਨ ਵਿੱਚ 18 ਮਹੀਨੇ ਲੱਗੇ ਅਤੇ ਇਸਦਾ ਸ਼ੁਰੂਆਤੀ ਨਾਮ ਓਕ (Oak) ਸੀ। ਪਰੰਤੂ ਕਾਪੀਰਾਈਟ ਮੁੱਦਿਆਂ ਕਾਰਨ 1995 ਵਿੱਚ ਓਕ (Oak) ਦਾ ਨਾਮ ਬਦਲ ਕੇ ਜਾਵਾ (Java) ਰੱਖਿਆ ਗਿਆ ਸੀ। ਜਾਵਾ ਦੀ ਪਹਿਲੀ ਜਨਤਕ ਸਥਾਪਨਾ (public implemention) JAVA 1.0 ਨਾਲ 1995 ਵਿੱਚ ਕੀਤੀ ਗਈ। ਜਾਵਾ ਕਾਫ਼ੀ ਸੁਰੱਖਿਅਤ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ।

ਜਾਵਾ ਇੰਡੋਨੇਸ਼ੀਆ ਵਿੱਚ ਇੱਕ ਟਾਪੂ ਦਾ ਨਾਮ ਹੈ ਜਿੱਥੇ ਪਹਿਲੀ ਕੌਫੀ (ਜਿਸਦਾ ਨਾਮ ਜਾਵਾ ਕੌਫੀ ਸੀ) ਪੈਦਾ ਕੀਤੀ ਗਈ ਸੀ; ਜੇਮਸ ਗੋਸਲਿੰਗ ਨੇ ਨਵੀਂ ਭਾਸ਼ਾ ਦਾ ਜਾਵਾ ਨਾਮ ਆਪਣੇ ਦਫਤਰ ਦੇ ਨੇੜੇ ਕੌਫੀ ਪੀਂਦੇ ਹੋਏ ਚੁਣਿਆ ਸੀ। ਜੇਮਸ ਗੋਸਲਿੰਗ ਨੂੰ ਜਾਵਾ ਦਾ ਪਿਤਾਮਾ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ॥

4.5 ਜਾਵਾ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (FEATURES OF JAVA) :

ਜਾਵਾ ਦੁਨੀਆ ਦੀ ਸਭ ਤੋਂ ਪ੍ਰਸਿੱਧ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਇਹ ਸਿੱਖਣ ਲਈ ਆਸਾਨ ਅਤੇ ਵਰਤਣ ਲਈ ਇਕ ਸਧਾਰਨ ਭਾਸ਼ਾ ਹੈ। ਇਸ ਵਿੱਚ ਬਹੁਤ ਸਾਰੀਆਂ ਅਜਿਹੀਆਂ ਸ਼ਕਤੀਸ਼ਾਲੀ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹਨ ਜੋ ਇਸਨੂੰ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਅਤੇ ਇੱਕ ਬਹੁਤ ਮਸ਼ਹੂਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਬਣਾਉਂਦੀਆਂ ਹਨ। ਹੇਠਾਂ ਜਾਵਾਂ ਦੀਆਂ ਕੁੱਝ ਮਹੱਤਵਪੂਰਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦਾ ਵਰਨਣ ਕੀਤਾ ਗਿਆ ਹੈ:

- **ਕੰਪਾਈਲਡ ਅਤੇ ਇੰਟਰਪ੍ਰੈਟਿਡ (Compiled and Interpretered) :** ਜ਼ਿਆਦਾਤਰ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ ਕਿ ਜਾਂ ਤਾਂ ਉਹ ਕੰਪਾਇਲਡ ਭਾਸ਼ਾਵਾਂ ਹਨ ਜਾਂ ਉਹ ਇੰਟਰਪ੍ਰੈਟਿਡ ਭਾਸ਼ਾਵਾਂ ਹਨ। ਜਾਵਾ ਕੋਲ ਕੰਪਾਈਲਡ ਅਤੇ ਇੰਟਰਪੇਟਿਡ ਦੋਵੇਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹਨ। ਜਾਵਾ ਕੰਪਾਈਲਰ ਸੋਰਸ ਕੋਡ ਨੂੰ ਬਾਈਟਕੋਡ ਵਿਚ ਕੰਪਾਇਲ ਕਰਦਾ ਹੈ ਅਤੇ JVM ਇਸ ਬਾਈਟਕੋਡ ਨੂੰ ਮਸ਼ੀਨ OS ਉਪਰ ਨਿਰਭਰ ਐਗਜ਼ੀਕਿਊਟੇਬਲ ਕੋਡ ਵਿਚ ਤਬਦੀਲ ਕਰਕੇ ਕਦਮ ਦਰ ਕਦਮ ਚਲਾਉਂਦਾ ਹੈ।
- **ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ (Platform Independent) :** ਜਾਵਾ ਭਾਸ਼ਾ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜਾਵਾ ਦਾ ਪ੍ਰੋਗਰਾਮ ਇੱਕ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਤੋਂ ਦੂਜੀ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਉਪਰ ਆਸਾਨੀ ਨਾਲ ਪ੍ਰਯੋਗ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਕਾਰਨ ਇਹ ਹੈ ਕਿ ਕੰਪਾਈਲਰ ਦੁਆਰਾ ਸੋਰਸ ਕੋਡ ਨੂੰ ਬਾਈਟਕੋਡ ਵਿੱਚ ਬਦਲਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਫਿਰ JVM ਦੁਆਰਾ ਇਸ ਬਾਈਟਕੋਡ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਕੋਡ ਵਿਚ ਤਬਦੀਲ ਕਰਕੇ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਬਾਈਟਕੋਡ ਕਿਸੇ ਵੀ ਪਲੇਟਫਾਰਮ 'ਤੇ ਚੱਲ ਸਕਦਾ ਹੈ ਭਾਵੇਂ ਉਹ ਵਿੰਡੋਜ਼, ਲੀਨਕਸ, ਜਾਂ ਮੈਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਹੋਵੇ, ਜਿਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜੇਕਰ ਅਸੀਂ ਵਿੰਡੋਜ਼ ਪਲੇਟਫਾਰਮ ਉਪਰ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਇਸਨੂੰ ਲੀਨਕਸ 'ਤੇ ਵੀ ਚਲਾ ਸਕਦੇ ਹਾਂ।
- **ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ (Object-Oriented) :** ਜਾਵਾ ਸ਼ੁੱਧ ਰੂਪ ਵਿਚ ਇਕ OOP ਭਾਸ਼ਾ ਹੈ। ਜਾਵਾ ਭਾਸ਼ਾ ਦਾ ਸਾਰਾ ਕੋਡ ਕਲਾਸਾਂ ਅਤੇ ਆਬਜੈਕਟਾਂ ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ OOP ਦੇ ਸਾਰੇ ਚਾਰ ਮੁੱਖ ਸੰਕਲਪਾਂ ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction), ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਇਨਹੇਰੀਟੈਂਸ (Inheritance), ਅਤੇ ਪੌਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।
- **ਮਜ਼ਬੂਤ (Robust) :** ਜਾਵਾ ਇਕ ਮਜ਼ਬੂਤ ਭਾਸ਼ਾ ਹੈ ਜਿਸਤੋਂ ਭਾਵ ਹੈ ਇਕ ਭਰੋਸੇਯੋਗ ਭਾਸ਼ਾ। ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਇਸ ਤਰੀਕੇ ਨਾਲ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਹੈ ਕਿ ਇਹ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਜਲਦੀ ਤੋਂ ਜਲਦੀ ਗਲਤੀਆਂ ਦੀ ਜਾਂਚ ਕਰ ਸਕੇ। ਜਾਵਾ ਕੰਪਾਈਲਰ ਉਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਵੀ ਲੱਭਣ ਦੇ ਯੋਗ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿਸੇ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੁਆਰਾ ਲੱਭਣਾ ਆਸਾਨ ਨਹੀਂ ਹੁੰਦਾ। ਗਾਰਬੇਜ਼ ਕੁਲੈਕਸ਼ਨ (garbage collection), ਐਕਸੇਪਸ਼ਨ ਹੈਂਡਲਿੰਗ (Exception Handling), ਅਤੇ ਮੈਮੋਰੀ ਐਲੋਕੇਸ਼ਨ (memory allocation) ਆਦਿ ਜਾਵਾ ਦੀਆਂ ਅਜਿਹੀਆਂ ਮੁੱਖ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹਨ ਜੋ ਇਸਨੂੰ ਮਜ਼ਬੂਤ ਬਣਾਉਂਦੀਆਂ ਹਨ।

❖ **ਸੁਰੱਖਿਅਤ (Secure) :** ਜਾਵਾ ਕਈ ਕਾਰਨਾਂ ਕਰਕੇ ਸੁਰੱਖਿਅਤ ਹੈ ॥ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਇੱਕ ਵਰਚੁਅਲ ਮਸ਼ੀਨ ਦੇ ਅੰਦਰ ਚੱਲਦੇ ਹਨ ਜਿਸਨੂੰ ਸੈਂਡਬਾਕਸ (Sandbox) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। JAVA, C ਭਾਸ਼ਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਪੁਆਇੰਟਰਜ਼ ਨੂੰ ਸਪੋਰਟ ਨਹੀਂ ਕਰਦਾ। ਬਾਈਟਕੋਡ ਵੈਰੀਫਾਇਰ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਉਹਨਾਂ ਗੈਰ ਕਾਨੂੰਨੀ ਕੋਡਜ਼ (illegal codes) ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ। ਜੋ ਆਬਜੈਕਟ ਦੇ ਅਧਿਕਾਰ ਦੀ ਉਲੰਘਣਾ (violate access right) ਕਰ ਸਕਦੇ ਹਨ। JAVA ਵਿਚ ਕਈ ਅਜਿਹੀਆਂ ਕਲਾਸ ਲਾਇਬ੍ਰੇਰੀਆਂ ਮੌਜੂਦ ਹਨ ਜੋ ਸੁਰੱਖਿਅਤ API ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ। ਇਹਨਾਂ API ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਰੇਕ ਅੰਦਰੂਨੀ ਸੰਚਾਰ ਦੀ ਪ੍ਰਮਾਣਿਕਤਾ ਦੀ ਜਾਂਚ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

❖ **ਡਿਸਟੀਬਿਊਟਿਡ (Distribulted) :** ਅਸੀਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਿਸਟ੍ਰੀਬਿਊਟਿਡ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾ ਸਕਦੇ ਹਾਂ। ।ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਅਜਿਹੇ ਇੱਕ ਜਾਂ ਵਧੇਰੇ ਸਿਸਟਮਾਂ 'ਤੇ ਆਸਾਨੀ ਨਾਲ ਵੰਡਿਆ (distribute) ਜਾ ਸਕਦਾ ਹੈ। ਜੋ ਇੰਟਰਨੈਟ ਕਨੈਕਸ਼ਨ ਦੁਆਰਾ ਇੱਕ ਦੂਜੇ ਨਾਲ ਜੁੜੇ ਹੁੰਦੇ ਹਨ। ਇਸ ਤਰ੍ਹਾਂ ਜਾਵਾ ਨੂੰ ਵਿਸ਼ੇਸ਼ ਤੌਰ 'ਤੇ ਉਹਨਾਂ ਇੰਟਰਨੈਟ-ਯੂਜ਼ਰਜ਼ ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਸੀ। ਜੋ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਰਿਮੋਟ ਕੰਪਿਊਟਰਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ। ।

❖ **ਸਧਾਰਨ, ਛੋਟੀ ਅਤੇ ਜਾਣੂ (Simple, Small and Familiar) :** ਜਾਵਾ ਇੱਕ ਸਧਾਰਨ ਭਾਸ਼ਾ ਹੈ ਕਿਉਂਕਿ ਇਸ ਵਿੱਚ C ਅਤੇ C++ ਵਰਗੀਆਂ ਹੋਰ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਬਹੁਤ ਸਾਰੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਸ਼ਾਮਲ ਹਨ। ਜਾਵਾ ਵਿਚ ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਗੁੰਝਲਦਾਰ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਜਿਵੇਂ ਕਿ: ਪੁਆਇੰਟਰਜ਼ (Pointers), ਸਟੋਰੇਜ਼ ਕਲਾਸਾਂ (Storge Classes) ਅਤੇ goto ਸਟੇਟਮੈਂਟ ਆਦਿ ਨੂੰ ਹਟਾਇਆ ਗਿਆ ਹੈ।ਇਹ ਮਲਟੀਪਲ ਇਨਹੇਰੀਟੈਂਸ (Inheritance), ਓਪਰੇਟਰ ਓਵਰਲੋਡਿੰਗ (Overloding), ਸਪੱਸ਼ਟ ਮੈਮੋਰੀ ਐਲੋਕੇਸ਼ਨ (Explicit Memory Allocation) ਆਦਿ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਸਪੋਰਟ ਨਹੀਂ ਕਰਦਾ ਜੋ C++ ਦੁਆਰਾ ਸਪੋਰਟ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।

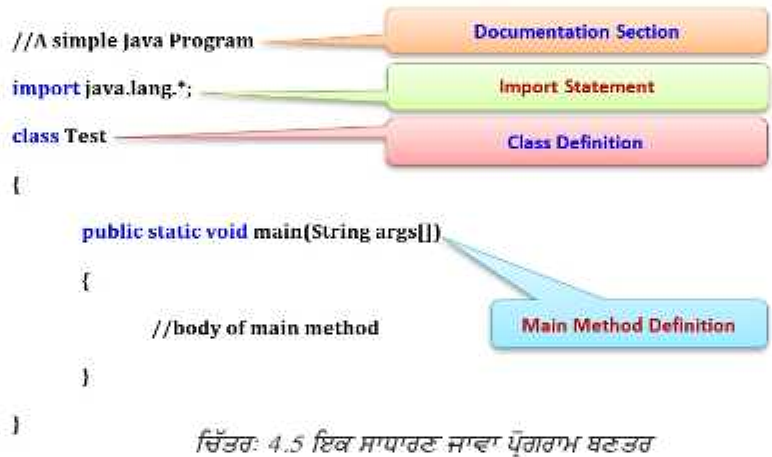
❖ **ਮਲਟੀਥਰੈਡਡ ਅਤੇ ਇੰਟਰਐਕਟਿਵ (Multithreaded and Interactive) :** ਜਾਵਾ ਮਲਟੀਥ੍ਰੈਡਿੰਗ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ ॥ ਇਹ ਇੱਕ ਅਜਿਹੀ ਜਾਵਾ ਵਿਸ਼ੇਸ਼ਤਾ ਹੈ ਜੋ CPU ਦੀ ਵੱਧ ਤੋਂ ਵੱਧ ਉਪਯੋਗਤਾ (Maximum Utilization) ਲਈ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਭਾਗਾਂ ਨੂੰ ਇੱਕੋ ਸਮੇਂ ਚਲਾਉਣ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ।

❖ **ਪੋਰਟੇਬਲ (Portable) :** ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਇਕ ਮਸ਼ੀਨ ਉਪਰ ਲਿਖਿਆ ਜਾਵਾ ਕੋਡ ਦੂਜੀ ਮਸ਼ੀਨ 'ਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਜਿਸਨੂੰ Write Once Run Anywhere-WORA ਵਿਸ਼ੇਸ਼ਤਾ ਨਾਲ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ ॥ ਜਾਵਾ ਦੀ ਪਲੇਟਫਾਰਮ-ਸੁਤੰਤਰ (Patfrom-Independent) ਵਿਸ਼ੇਸ਼ਤਾ ਜਿਸ ਵਿੱਚ ਇਸਦੇ ਬਾਈਟਕੋਡ (Bytecode) ਨੂੰ ਐਗਜ਼ੀਕਿਊਸ਼ਨ (exection) ਲਈ ਕਿਸੇ ਵੀ ਪਲੇਟਫਾਰਮ 'ਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ, ਜਾਵਾ ਨੂੰ ਪੋਰਟੇਬਲ ਬਣਾਉਂਦੀ ਹੈ ॥

❖ **ਡਾਇਨਾਮਿਕ ਅਤੇ ਐਕਸਟੈਂਸੀਬਲ ਕੋਡ (Dynamic and Extensible Code) :** ਜਾਵਾ ਪੂਰੀ ਤਰ੍ਹਾਂ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਹੋਣ ਕਾਰਨ ਸਾਨੂੰ ਨਵੀਆਂ ਕਲਾਸਾਂ, ਮੌਜੂਦਾ ਕਲਾਸਾਂ ਵਿੱਚ ਨਵੇਂ ਮੈਥਡਜ਼ (Methods) ਅਤੇ ਸਬ-ਕਲਾਸਾਂ (sub-classes) ਦੀ ਵਰਤੋਂ ਨਾਲ ਨਵੀਆਂ ਕਲਾਸਾਂ ਬਣਾਉਣ ਦੀ ਲਚਕਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। JAVA ਹੋਰ ਭਾਸ਼ਾਵਾਂ ਜਿਵੇਂ ਕਿ C, C++ ਵਿੱਚ ਲਿਖੇ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਵੀ ਸਪੋਰਟ ਕਰਦਾ ਹੈ ਜਿਨ੍ਹਾਂ ਨੂੰ ਨੇਟਿਵ ਮੈਥਡਜ਼ (Native Methods) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

4.6 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਮੁੱਢਲੀ ਬਣਤਰ (BASIC STRUCTURE OFJAVA PROGRAM)

ਹਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਿੱਚ ਕੁੱਝ ਪਹਿਲਾਂ ਤੋਂ ਪ੍ਰਭਾਸ਼ਿਤ ਕੋਡ ਬਣਤਰ ਅਤੇ ਸਿੰਟੈਕਸ ਵਜੋਂ ਜਾਣੇ ਜਾਂਦੇ ਨਿਯਮਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦਾ ਹੈ। ਜੇਕਰ ਕੋਡ ਲਿਖਣ ਵੇਲੇ ਇਹਨਾਂ ਨਿਯਮਾਂ ਦੀ ਉਲੰਘਣਾ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਤਾਂ ਇਹ ਇੱਕ ਐਰਰ (Error) ਦਰਸ਼ਾਵੇਗਾ। ਹਰੇਕ ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ ਦੀ ਪਾਲਣਾ ਕਰਨਾ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਹੁੰਦਾ ਹੈ। ਇਸ ਤੋਂ ਪਹਿਲਾਂ ਕਿ ਅਸੀਂ ਇਹ ਸਿੱਖਣਾ ਸ਼ੁਰੂ ਕਰੀਏ ਕਿ ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਕਿਵੇਂ ਲਿਖਣਾ ਹੈ, ਆਓ ਪਹਿਲਾਂ ਇਹ ਸਮਝੀਏ ਕਿ ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦਾ ਬੁਨਿਆਦੀ ਢਾਂਚਾ ਕੀ ਹੁੰਦਾ ਹੈ।



ਚਿੱਤਰ: 4.5 ਇਕ ਸਾਧਾਰਣ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਤਰ

ਇਹ ਚਿੱਤਰ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਬਹੁਤ ਹੀ ਬੁਨਿਆਦੀ ਬਣਤਰ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ।ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਹੋਰ ਤੱਤ (elements) ਵੀ ਹੋ ਸਕਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਪੈਕੇਜ ਸਟੇਟਮੈਂਟ (package statement), ਇੰਟਰਫੇਸ ਸੈਕਸ਼ਨ (Inerface section) ਆਦਿ॥ ਅਸੀਂ ਇਹਨਾਂ ਭਾਗਾਂ ਬਾਰੇ ਚਰਚਾ ਨਹੀਂ ਕਰਾਂਗੇ ਕਿਉਂਕਿ ਇਹ ਟੌਪਿਕਸ ਇਸ ਕਿਤਾਬ ਦੇ ਦਾਇਰੇ (scope) ਤੋਂ ਬਾਹਰ ਹਨ॥ ਆਓ ਹੁਣ ਅਸੀਂ ਉਪਰੋਕਤ ਪ੍ਰੋਗਰਾਮ-ਸਟ੍ਰਕਚਰ ਵਿੱਚ ਦਰਸਾਏ ਪ੍ਰੋਗਰਾਮ ਤੱਤਾਂ ਦੇ ਵਰਣਨ 'ਤੇ ਇੱਕ ਨਜ਼ਰ ਮਾਰੀਏ:

- **ਡਾਕੂਮੈਂਟੇਸ਼ਨ ਸੈਕਸ਼ਨ (Documentation Section) :** ਇਹ ਪ੍ਰੋਗਰਾਮ ਦੀ ਪੜ੍ਹਨਯੋਗਤਾ (readability) ਨੂੰ ਬਿਹਤਰ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। । ਇਸ ਵਿੱਚ JAVA ਦੇ ਕਮੈਂਟਸ ਸ਼ਾਮਿਲ ਹੁੰਦੇ ਹਨ। ਜੋ ਕੋਡ ਦੀ ਸਮੀਖਿਆ (reviewing) ਜਾਂ ਡੀਬੱਗਿੰਗ (debugging) ਕਰਨ ਵੇਲੇ ਪ੍ਰੋਗਰਾਮਰ ਦੇ ਕੰਮ ਨੂੰ ਆਸਾਨ ਬਣਾਉਂਦੇ ਹਨ॥ ਇਹ ਸਟੇਟਮੈਂਟਸ ਆਪਸ਼ਨਲ ਹੁੰਦੀਆਂ ਹਨ। :
- **ਇੰਪੋਰਟ ਸਟੇਟਮੈਂਟ (Impornt Statement) :** ਜਾਵਾ ਵਿੱਚ ਬਹੁਤ ਸਾਰੀਆਂ ਪੂਰਵ-ਪ੍ਰਭਾਸ਼ਿਤ (predefined) ਕਲਾਸਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਪੈਕੇਜਾਂ (packages) ਦੇ ਰੂਪ ਵਿਚ ਸੰਗਠਿਤ (organized) ਹੁੰਦੀਆਂ ਹਨ। ਜੇਕਰ ਅਸੀਂ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਦੂਜੇ ਪੈਕੇਜਾਂ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕਲਾਸਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਤਾਂ ਸਾਨੂੰ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ import ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਕਰਨੀ ਪਵੇਗੀ।
- **ਕਲਾਸ ਪਰਿਭਾਸ਼ਾ (Class Defination) :** ਕਲਾਸਾਂ ਕਿਸੇ ਵੀ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦਾ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਹੁੰਦੀਆਂ ਹਨ। ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਈ ਕਲਾਸ ਪਰਿਭਾਸ਼ਾਵਾਂ ਹੋ ਸਕਦੀਆਂ ਹਨ। JAVA ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣ ਲਈ ਮੇਨ main ਮੈਥਡ ਵਾਲੀ ਇੱਕ ਕਲਾਸ ਬਨਾਉਣੀ ਜਰੂਰੀ ਹੁੰਦੀ ਹੈ॥ਜਿਸ ਕਲਾਸ ਵਿੱਚ ਮੇਨ ਮੈਥਡ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ ਉਸਨੂੰ ਪਬਲਿਕ (public) ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ।
  - **class Test -** ਜਾਵਾ ਵਿਚ ਇਹ ਸਟੇਟਮੈਂਟ ਇੱਕ ਕਲਾਸ ਬਣਾਵੇਗੀ ਜਿਸਦਾ ਨਾਮ Test ਹੋਵੇਗਾ॥ ਜਾਵਾ ਦੇ ਸਟੈਂਡਰਡ ਕਨਵੈਨਸ਼ਨ ਅਨੁਸਾਰ ਕਲਾਸ ਦੇ ਨਾਮ ਲਈ ਟਾਈਟਲਕੇਸ (Title Case) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ॥ ਇਸ ਲਈ ਜਾਵਾ ਵਿਚ ਕਲਾਸ ਦਾ ਨਾਮ ਤੈਅ ਕਰਦੇ ਹੋਏ ਇਹ ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਕਲਾਸ ਦਾ ਨਾਮ ਟਾਈਟਲਕੇਸ (ਨਾਮ ਵਿੱਚ ਖਾਲੀ ਥਾਂ ਨਹੀਂ ਹੋਣੀ ਚਾਹੀਦੀ) ਵਿੱਚ ਹੋਵੇ।
  - **ਬਰੇਸਿਸ (Braces{}) (ਓਪਨਿੰਗ ਅਤੇ ਕਲੋਜ਼ਿੰਗ ਦੋਵੇਂ ਤਰ੍ਹਾਂ ਦੇ) -** ਕਰਲੀ ਬਰੇਸਿਸ ({ }) ਦੀ ਵਰਤੋਂ ਕਈ ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਇੱਕ ਗਰੁੱਪ ਵਿਚ ਇੱਕਠਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ॥ਕਰਲੀ ਬਰੇਸਿਸ ਦੀ ਵਰਤੋਂ ਇਹ ਦਰਸ਼ਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਦਿੱਤੀਆਂ ਗਈਆਂ ਸਟੇਟਮੈਂਟਸ ਕਲਾਸ ਜਾਂ ਮੈਥਡ ਨਾਲ ਸਬੰਧਤ ਹਨ॥
  - **main() ਮੈਥਡ ਪਰਿਭਾਸ਼ਾ (main () Method Defintion) :** ਜਾਵਾ ਵਿੱਚ ਮੇਨ ਮੈਥਡ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਂਟਰੀ ਪੁਆਇੰਟ ਵਜੋਂ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ॥ ਜਦੋਂ ਯੂਜ਼ਰ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਂਦਾ ਹੈ ਤਾਂ ਮੇਨ ਮੈਥਡ ਨੂੰ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੁਆਰਾ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ: ਉਦਾਹਰਣ ਲਈ:

```
public static void main(String args[])
```

- **public -** ਜਦੋਂ ਮੇਨ (main( )) ਮੈਥਡ ਨੂੰ ਪਬਲਿਕ (public) ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸਦਾ ਮਤਲਬ ਹੁੰਦਾ ਹੈ ਕਿ ਇਸਨੂੰ ਘੋਸ਼ਿਤ ਕਲਾਸ (declared class) ਤੋਂ ਬਾਹਰ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

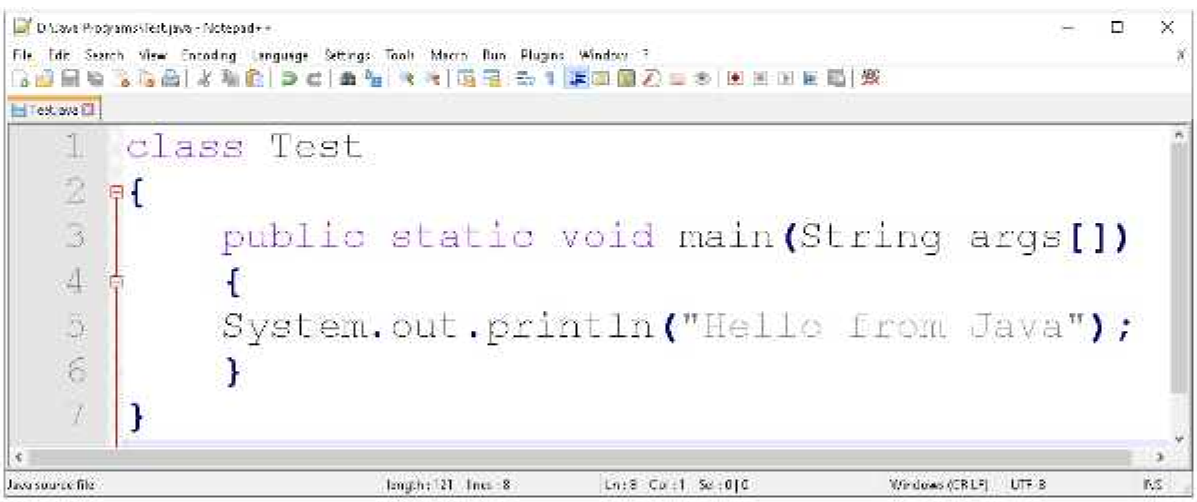
- **static** – ਸਟੈਟਿਕ ਸ਼ਬਦ ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਅਸੀਂ ਉਸ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਏ ਬਿਨਾਂ ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ ਜਿਸ ਵਿੱਚ ਮੈਥਡ ਘੋਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ॥ਜਾਵਾ ਵਿਚ ਅਸੀਂ ਮੇਨ ਮੈਥਡ ਵਾਲੀ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਏ ਬਿਨਾਂ ਮੇਨ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕਰਦੇ ਹਾਂ ਜਿਸ ਕਾਰਨ ਮੇਨ ਮੈਥਡ ਨਾਲ Static ਕੀਅਵਰਡ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
- **void** – ਕਿਸੇ ਵੀ ਮੈਥਡ ਨਾਲ void ਸ਼ਬਦ ਇਹ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ ਉਹ ਮੈਥਡ ਕੋਈ ਵੀ ਮੁੱਲ ਯੂਜ਼ਰ ਨੂੰ ਵਾਪਿਸ ਨਹੀਂ ਕਰੇਗਾ।ਮੇਨ ਮੈਥਡ ਨੂੰ ਇਸੇ ਕਰਕੇ void ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਉਹ ਕੋਈ ਵੀ ਮੁੱਲ ਯੂਜ਼ਰ ਨੂੰ ਵਾਪਸ ਨਹੀਂ (does not return any value) ਕਰਦਾ।
- **main** - ਇਹ ਇਕ ਮੈਥਡ ਹੈ, ਜੋ ਕਿ ਕਿਸੇ ਵੀ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦਾ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਹੁੰਦਾ ਹੈ।
- **String args[ ]** - ਇਹ ਇੱਕ ਐਰੇ (array) ਹੈ ਜਿੱਥੇ ਹਰੇਕ ਤੱਤ (element) ਇੱਕ ਸਟਿੰਗ ਹੁੰਦਾ ਹੈ, ਜਿਸਨੂੰ args ਨਾਮ ਦਿਤਾ ਗਿਆ ਹੈ। ।ਜੇਕਰ ਅਸੀਂ ਜਾਵਾ ਕੋਡ ਨੂੰ ਕੰਸੋਲ (console) ਰਾਹੀਂ ਚਲਾਉਂਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਇਸ ਐਰੇ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇਨਪੁਟ ਪੈਰਾਮੀਟਰਜ਼ ਪਾਸ ਕਰ ਸਕਦੇ ਹਾਂ।main() ਮੈਥਡ ਇਸਨੂੰ ਇੱਕ ਇਨਪੁੱਟ ਵਜੋਂ ਲੈਂਦਾ ਹੈ।

4.7 ਇੱਕ ਸਧਾਰਨ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣਾ ਅਤੇ ਲਾਗੂ ਕਰਨਾ (CREATING AND EXECUTING A SIMPLE JAVA PROGRAM)

ਆਪਣਾ ਪਹਿਲਾ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣ ਲਈ ਤੁਹਾਨੂੰ ਹੇਠਾਂ ਦਿੱਤੇ ਦੋ ਸਾਫਟਵੇਅਰਾਂ ਦੀ ਜਰੂਰਤ ਪਵੇਗੀ:

1. **ਜਾਵਾ ਡਿਵੇਲਪਮੈਂਟ ਕਿੱਟ (Java Development Kit (JDK))** : ਇਹ JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਮੁੱਖ ਪਲੇਟਫਾਰਮ ਕੰਪੋਨੈਂਟ ਹੈ। ।ਇਸਦਾ ਮੁੱਖ ਭਾਗ ਜਾਵਾ ਕੰਪਾਈਲਰ ਹੁੰਦਾ ਹੈ। ਇਸ JRE + ਡਿਵੈਲਪਮੈਂਟ ਟੂਲਜ਼ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ॥
2. **ਟੈਕਸਟ ਐਡੀਟਰ (Text Editor)** : ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਦਾ ਸੋਰਸ ਕੋਡ ਲਿਖਣ ਲਈ ਕਿਸੇ ਵੀ ਟੈਕਸਟ ਐਡੀਟਰ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ Notepad, Notepad+ + ਆਦਿ। ਨਿਮਨਲਿਖਤ ਪ੍ਰੋਗਰਾਮ ਦੀ ਉਦਾਹਰਨ ਵਿੱਚ ਅਸੀਂ Notepad ++ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹਾਂ। ।ਅਸੀਂ ਕਿਸੇ ਹੋਰ ਵੱਖਰੇ ਟੈਕਸਟ ਐਡੀਟਰ ਦੀ ਵਰਤੋਂ ਵੀ ਕਰ ਸਕਦੇ ਹਾਂ। Notepad ਵਿੰਡੋਜ਼ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਮੌਜੂਦ ਸਧਾਰਨ ਟੈਕਸਟ-ਐਡੀਟਰ ਹੁੰਦਾ ਹੈ, ਜਾਂ ਆਨਲਾਈਨ ਜਾਵਾ ਕੰਪਾਈਲਰ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣ ਅਤੇ ਚਲਾਉਣ ਲਈ ਹੇਠਾਂ ਦਿੱਤੇ ਸਟੈਪ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ:

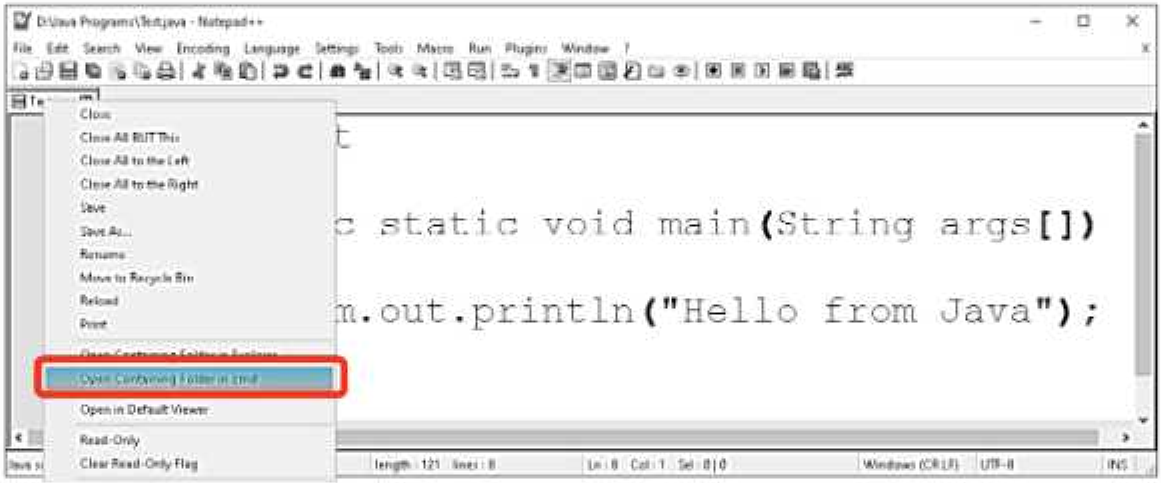
ਸਟੈਪ 1: Notepad+ + ਓਪਨ ਕਰੋ (Start button, Notepad++) ਅਤੇ ਉਸ ਵਿਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਟਾਈਪ ਕਰੋ, ਫਿਰ ਉਸਨੂੰ **Java** ਐਕਸਟੈਂਸ਼ਨ (extension) ਨਾਲ ਸੇਵ (Ctrl+S) ਕਰੋ, ਉਦਾਹਰਣ ਲਈ:



ਚਿੱਤਰ 4.6 JAVA ਪ੍ਰੋਗਰਾਮ ਦੀ Notepad++ ਵਿੱਚ ਉਦਾਹਰਣ

ਸਟੈਪ 2: ਮੌਜੂਦਾ ਫਾਈਲ ਦੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਕਮਾਂਡ ਪ੍ਰਾਮਪਟ ਉਪਰ ਖੋਲਣ ਲਈ Notepad+ + ਦੇ ਮੌਜੂਦਾ ਫਾਈਲ ਟੈਬ ਉੱਪਰ Right

ਕਲਿੱਕ ਕਰੋ ਅਤੇ ਫਿਰ ਓਪਨ ਹੋਏ ਪ੍ਰਾਪਰਟੀਜ਼ ਮੀਨੂੰ ਵਿਚੋਂ **"Open Containing Folder in CMD** ਆਪਸ਼ਨ ਉਪਰ ਕਲਿੱਕ ਕਰੋ, ਜਿਵੇਂ ਕਿ ਅੱਗੇ ਚਿੱਤਰ ਵਿਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:



ਚਿੱਤਰ: 4.7 Notepad+ + ਵਿਚ ਮੌਜੂਦਾ ਫਾਈਲ ਲਈ ਕਮਾਂਡ ਪ੍ਰੋਮਪਟ ਖੋਲਣਾ

ਚਿੱਤਰ: 4.8 ਮੌਜੂਦਾ ਫਾਈਲ ਦੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਕਮਾਂਡ ਪ੍ਰਾਮਪਟ

ਸਟੈੱਪ 3: ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਨ ਲਈ ਹੇਠਾਂ ਦਿਤੇ ਕਮਾਂਡ ਸਿੰਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕਰੋ:

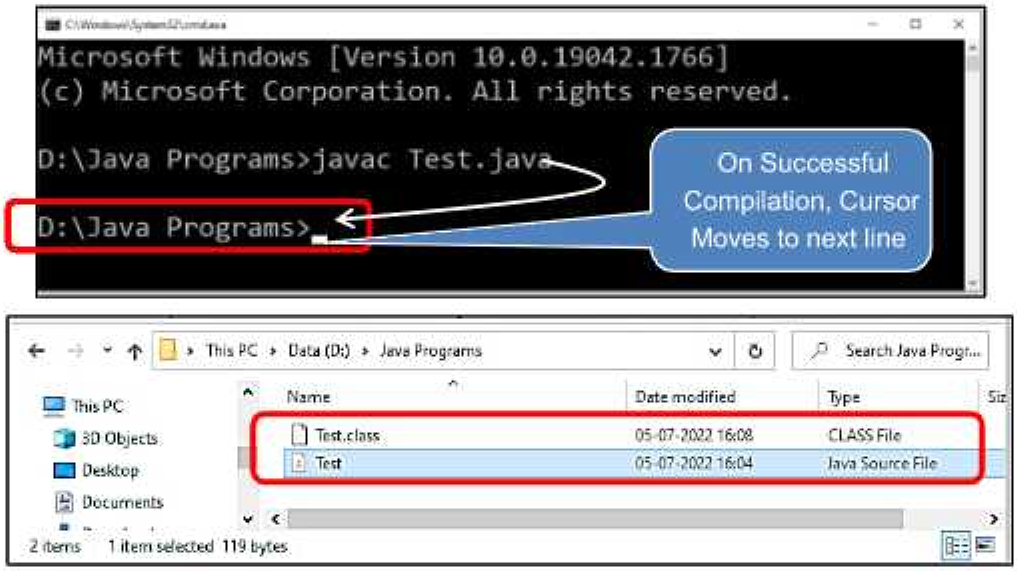
javac filename.java

ਹੇਠਾਂ ਦਰਸ਼ਾਏ ਅਨੁਸਾਰ Test.java ਫਾਈਲ ਨੂੰ ਕੰਪਾਈਲ ਕਰੋ:

ਚਿੱਤਰ 4.9: ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ Test.java ਨੂੰ ਕੰਪਾਈਲ ਕਰਨਾ

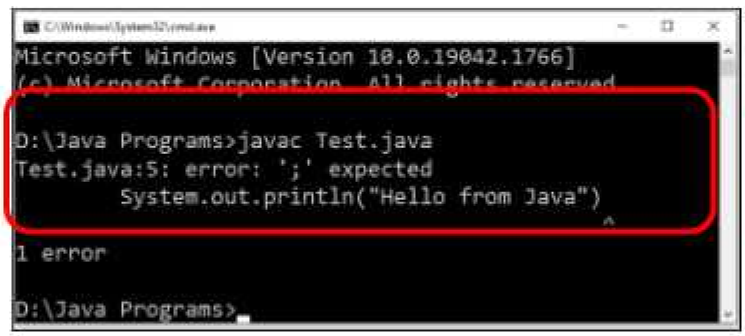
ਕੰਸੋਲ (Console) ਸਕੀਨ ਉਪਰ JAVA Test.java ਟਾਈਪ ਕਰਨ ਤੋਂ ਬਾਅਦ ਐਂਟਰ ਕੀਅ ਦਬਾਓ ॥ ਇਹ ਕਮਾਂਡ Test.java ਫਾਈਲ ਵਿੱਚ ਸਟੋਰ ਸਾਡੇ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੇ ਸੋਰਸ ਕੋਡ ਨੂੰ ਕੰਪਾਇਲ ਕਰੇਗਾ। ਜੇਕਰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੋਈ ਗਲਤੀ ਨਹੀਂ ਹੁੰਦੀ, ਤਾਂ ਇਹ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦਰਸਾਏ ਅਨੁਸਾਰ ਕਮਾਂਡ ਪ੍ਰੋਮਪਟ ਨੂੰ ਅਗਲੀ ਲਾਈਨ ਵਿੱਚ ਦਿਖਾਏਗਾ।

ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਈਲ ਹੋਣ ਤੋਂ ਬਾਅਦ ਇਹ ਉਸੇ ਫੋਲਡਰ ਵਿੱਚ ਇੱਕ .class ਫਾਈਲ ਤਿਆਰ ਕਰ ਦੇਵੇਗਾ। ਜਿਸ ਵਿੱਚ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਦਾ ਬਾਈਟਕੋਡ ਸਟੋਰ ਹੋ ਜਾਵੇਗਾ॥ ਇਸ ਬਾਈਟਕੋਡ ਨੂੰ ਹੀ ਜਾਵਾ ਇੰਟਰਪ੍ਰੇਟਰ ਦੁਆਰਾ JVM ਵਿੱਚ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ।



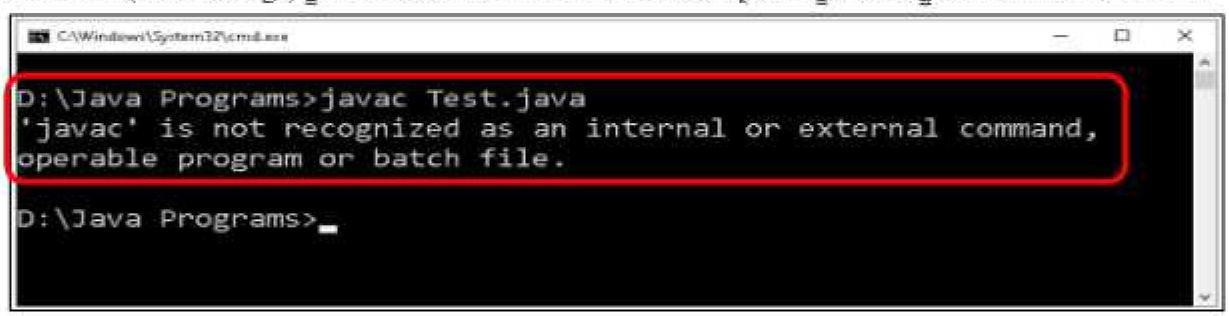
ਚਿੱਤਰ : 4.10 ਫਾਈਲ ਦੇ ਕੰਪਾਈਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ Test.class ਤਿਆਰ ਹੋ ਜਾਵੇਗੀ।

ਪਰ ਜੇਕਰ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੁੱਝ ਸਿੰਟੈਕਸ ਐਰਰਜ਼ (syntax errors) ਹੋਣ ਤਾਂ ਇਹ ਸੋਰਸ ਕੋਡ ਫਾਈਲ ਵਿਚ ਮੌਜੂਦ ਐਰਰਜ਼ ਨਾਲ ਸੰਬੰਧਤ ਮੈਸੇਜ, ਸਮੇਤ ਲਾਈਨ ਨੰਬਰ ਦਿਖਾਏਗਾ; ਜਿਵੇਂ ਕਿ ਹੇਠਾਂ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:



ਚਿੱਤਰ 4.11 ਕੰਪਾਇਲੇਸ਼ਨ ਐਰਰਜ਼ (Compilition Erros) (ਜੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਮੌਜੂਦ ਹੋਣ)

ਜੇਕਰ ਸਾਨੂੰ ਕੰਸੋਲ ਸਕੀਨ ਉਪਰ ਹੇਠਾਂ ਦਿੱਤਾ ਐਰਰ ਮੈਸੇਜ (Error Message) ਦਿਖਾਈ ਦੇ ਰਿਹਾ ਹੈ, ਤਾਂ ਸਾਨੂੰ ਆਪਣੇ ਸਿਸਟਮ ਦੀਆਂ ਪਾਥ ਸੈਟਿੰਗਾਂ (Path Settings) ਨੂੰ ਐਡਿਟ ਕਰਨਾ ਪਵੇਗਾ। ਤਾਂ ਜੋ ਅਸੀਂ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲ ਕਰ ਸਕੀਏ।



ਚਿੱਤਰ : 4.12 JAVA ਇੰਸਟਾਲੇਸ਼ਨ ਨਾਲ ਸੰਬਧਤ ਪਾਥ ਸੈਟਿੰਗ ਐਰਰ

ਸਟੈਪ 4: JAVA ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣ ਲਈ ਹੇਠ ਦਿੱਤੇ ਕਮਾਂਡ ਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕਰੋ:

Syntax:

java classname (ਨੋਟ: ਕਲਾਸ ਦਾ ਨਾਮ ਲਿਖਦੇ ਸਮੇਂ ਕੇਸਿੰਗ (CASEING) ਦਾ ਧਿਆਨ ਰੱਖੋ)

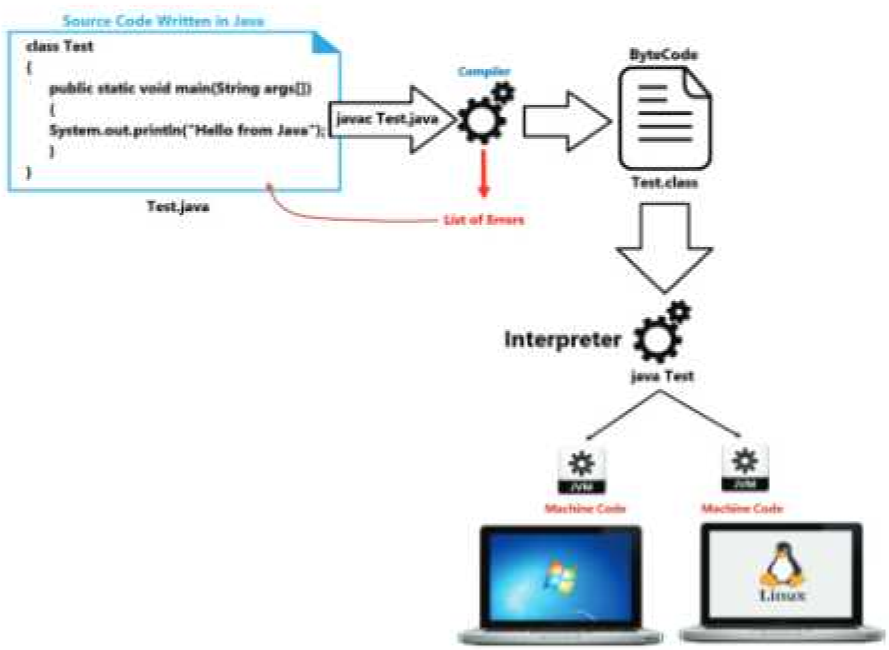
ਅੱਗੇ ਦਿੱਤਾ ਚਿੱਤਰ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ Test.Java ਦੀ ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ ਉਸ ਨੂੰ ਚਲਾਉਣ (execute) ਸੰਬੰਧੀ ਕਮਾਂਡ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਚਿੱਤਰ 4.13: ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣਾ

ਕੰਸੋਲ ਸਕ੍ਰੀਨ ਉਪਰ java Test ਟਾਈਪ ਕਰਨ ਤੋਂ ਬਾਅਦ ਕੀਬੋਰਡ ਤੋਂ ਐਂਟਰ ਕੀਅ ਦਬਾਓ ਅਤੇ ਇਹ ਹੇਠਾਂ ਦਰਸਾਏ ਅਨੁਸਾਰ ਪ੍ਰੋਗਰਾਮ ਦੀ ਆਉਟਪੁੱਟ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰੇਗਾ:

ਚਿੱਤਰ: 4.14: Test.java ਪ੍ਰੋਗਰਾਮ ਦੀ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਤੋਂ ਬਾਅਦ ਆਉਟਪੁੱਟ

ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕਰਨ ਅਤੇ ਉਸਨੂੰ ਚਲਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:

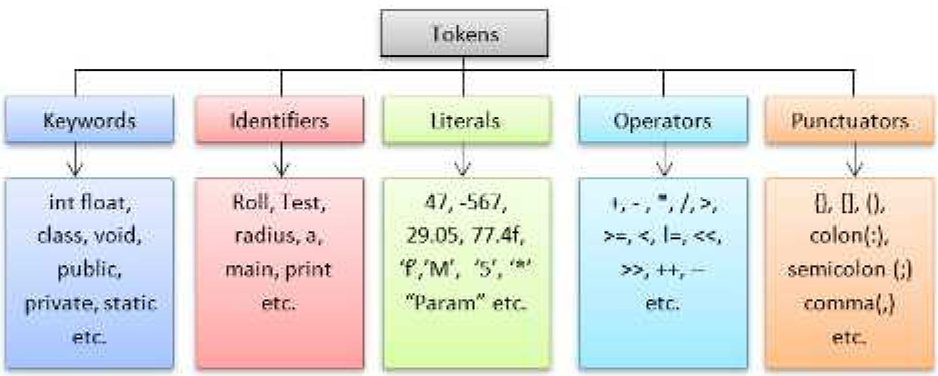


ਚਿੱਤਰ: 4.15 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ ਅਤੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਪ੍ਰੋਸੈਸ

4.8 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਬੁਨਿਆਦੀ ਤੱਤ (Basic Elements for Java Programming) : ਮਨੁੱਖਾਂ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਅਸੀਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾਵਾਂ, ਜਿਵੇਂ ਕਿ ਅੰਗਰੇਜ਼ੀ, ਹਿੰਦੀ ਅਤੇ ਪੰਜਾਬੀ ਆਦਿ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਪਰ ਕੰਪਿਊਟਰਾਂ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਅਸੀਂ ਸਿਰਫ ਉਹੀ ਭਾਸ਼ਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ ਜਿਹੜੀਆਂ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੁਆਰਾ ਸਿੱਧੇ ਜਾਂ ਅਸਿੱਧੇ ਢੰਗ ਨਾਲ ਸਮਝੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਹੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਇੱਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ। ਜਿਵੇਂ ਕਿ ਸਾਨੂੰ ਕਿਸੇ ਵੀ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਇਸਨੂੰ ਸਿੱਖਣਾ ਪੈਂਦਾ ਹੈ, ਠੀਕ ਉਸੇ ਤਰ੍ਹਾਂ ਸਾਨੂੰ ਕੰਪਿਊਟਰਾਂ ਨਾਲ ਸੰਚਾਰ (**communication**) ਕਰਨ ਲਈ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵੀ ਸਿੱਖਣੀਆਂ ਪੈਂਦੀਆਂ ਹਨ। ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਸਿੱਖਣਾ ਵੀ ਕਿਸੇ ਵੀ ਕੁਦਰਤੀ ਭਾਸ਼ਾ, ਜਿਵੇਂ ਹਿੰਦੀ, ਪੰਜਾਬੀ ਆਦਿ, ਨੂੰ ਸਿੱਖਣ ਵਾਂਗ ਹੀ ਹੁੰਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਕਿਵੇਂ ਲਿਖਣਾ ਹੈ ਇਹ ਸਿੱਧਾ ਸਿੱਖਣ ਦੀ ਬਜਾਏ ਸਾਨੂੰ ਪਹਿਲਾਂ ਇਹ ਪਤਾ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਕਿ ਜਾਵਾ ਵਿੱਚ ਕਿਹੜੇ ਅੱਖਰ, ਨੰਬਰ ਅਤੇ ਚਿੰਨ੍ਹ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਫਿਰ ਇਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਵੱਖ ਵੱਖ ਟੋਕਨਾਂ ਨੂੰ ਕਿਵੇਂ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਅੰਤ ਵਿੱਚ ਇਹਨਾਂ ਟੋਕਨਾ ਨੂੰ ਜੋੜ ਕੇ ਹਦਾਇਤਾਂ ਨੂੰ ਕਿਵੇਂ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਹਦਾਇਤਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਆਖਿਰ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਨ ਲਈ ਜੋੜਿਆ ਜਾਵੇਗਾ। ਆਉ ਹੁਣ ਜਾਵਾ ਦੇ ਕਰੈਕਟਰ ਸੈੱਟ ਨੂੰ ਸਮਝਦੇ ਹੋਏ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਨੂੰ ਸਿੱਖਣ ਦਾ ਪ੍ਰੋਸੈਸ ਸ਼ੁਰੂ ਕਰੀਏ।

4.8.1 ਕਰੈਕਟਰ ਸੈੱਟ (Character Set) : ਇਹ ਕਿਸੇ ਵੀ ਭਾਸ਼ਾ, ਚਾਹੇ ਉਹ ਕੁਦਰਤੀ ਹੋਵੇ ਜਾਂ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ, ਨੂੰ ਸਿੱਖਣ ਦਾ ਪਹਿਲਾ ਸਟੈਪ ਹੈ।ਅਸੀਂ ਕੋਈ ਵੀ ਭਾਸ਼ਾ ਤਾਂ ਹੀ ਸਿੱਖ ਸਕਦੇ ਹਾਂ ਜੇ ਅਸੀਂ ਉਸ ਭਾਸ਼ਾ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਅੱਖਰਾਂ ਅਤੇ ਚਿੰਨ੍ਹਾਂ (characters and symbols) ਦੀ ਜਾਣਕਾਰੀ ਰੱਖਦੇ ਹਾਂ। ਇਸਲਈ ਜਾਵਾ ਭਾਸ਼ਾ ਸਿੱਖਣ ਤੋਂ ਪਹਿਲਾਂ ਸਾਨੂੰ ਇਸ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਕਰੈਕਟਰਾਂ ਅਤੇ ਚਿੰਨ੍ਹਾਂ ਤੋਂ ਜਾਣੂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ॥C ਅਤੇ C++ ਵਰਗੀਆਂ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੇ ਉਲਟ, ਜਾਵਾ ਵਿਚ ਅੱਖਰਾਂ (ਚਹਓਰਓਚਟਏਰਸ) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਯੂਨੀਕੋਡ (Unicode) ਸਟੈਂਡਰਡ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਯੂਨੀਕੋਡ ਇੱਕ 16-ਬਿੱਟ ਕਰੈਕਟਰ ਕੋਡ ਸੈੱਟ ਹੈ। ਜੋ ਸਾਰੀਆਂ ਮਨੁੱਖੀ ਭਾਸ਼ਾਵਾਂ ਜਿਵੇਂ ਕਿ: ਅੰਗਰੇਜ਼ੀ, ਹਿੰਦੀ, ਫ੍ਰੈਂਚ, ਜਰਮਨ, ਚੀਨੀ, ਜਾਪਾਨੀ ਆਦਿ ਵਿੱਚ ਉਪਲਬਧ ਸਾਰੇ ਅੱਖਰਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ। ਯੂਨੀਕੋਡ ਕਰੈਕਟਰ ਸੈੱਟ ਜਾਵਾ ਨੂੰ ਇੱਕ ਵਿਸ਼ਵਵਿਆਪੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਬਣਾਉਂਦਾ ਹੈ। ।

4.8.2 ਟੋਕਨਜ਼ (Tokens) : ਟੋਕਨ ਅੰਗਰੇਜ਼ੀ ਭਾਸ਼ਾ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਸ਼ਬਦਾਂ ਅਤੇ ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹਾਂ Punchuation Marks ਵਾਂਗ ਹੁੰਦੇ ਹਨ। ਕਿਸੇ ਵੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਿੱਚ, ਜਿਵੇਂ ਕਿ ਜਾਵਾ ਭਾਸ਼ਾ ਵਿੱਚ, ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਟੋਕਨਾਂ ਦਾ ਬਣਿਆ ਹੁੰਦਾ ਹੈ। ਟੋਕਨ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਭ ਤੋਂ ਛੋਟੀਆਂ ਵਿਅਕਤੀਗਤ ਇਕਾਈਆਂ Small Indivldual Units ਹੁੰਦੀਆਂ ਹਨ। ਜਦੋਂ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਕੰਪਾਈਲਰ ਸੋਰਸ ਕੋਡ ਨੂੰ ਸਕੈਨ ਕਰਦਾ ਹੈ ਅਤੇ ਸਿੰਟੈਕਸ ਐਰਰਜ਼ ਨੂੰ ਲੱਭਣ ਲਈ ਸੋਰਸ ਕੋਡ ਨੂੰ ਟੋਕਨਾਂ ਵਿੱਚ ਪਾਰਸ (Parse) ਕਰਦਾ ਹੈ। ਜਾਵਾ ਟੋਕਨਾਂ ਨੂੰ ਮੋਟੇ ਤੌਰ 'ਤੇ ਕੀਵਰਡਸ, ਆਈਡੈਂਟੀਫਾਇਰਜ਼, ਲਿਟਰਲਜ਼ ਕਾਂਸਟੈਂਟਸ), C ਅਤੇ ਪੰਕਚੁਏਟਰਜ਼ (ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹ) ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਗਿਆ ਹੈ:



ਚਿੱਤਰ: 4.16 ਜਾਵਾ ਵਿਚ ਟੋਕਨਜ਼ ਦੀਆਂ ਕਿਸਮਾਂ

4.8.2.1 ਕੀਵਰਡਜ਼ (Keywords) : ਕੀਵਰਡਜ਼ ਨੂੰ ਰਿਜ਼ਰਵ ਵਰਡਜ਼ (Reserve Words) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸ਼ਬਦ ਜਾਵਾ ਕੰਪਾਈਲਰ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਪਰਿਭਾਸ਼ਿਤ ਹੁੰਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਸ਼ਬਦਾਂ ਦੇ ਅਰਥ ਪਹਿਲਾਂ ਤੋਂ ਪਰਿਭਾਸ਼ਿਤ ਹੁੰਦੇ ਹਨ।ਕੀਵਰਡਜ਼ ਨੂੰ ਉਹਨਾਂ ਵਿਸ਼ੇਸ਼ ਉਦੇਸ਼ਾਂ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਜਿਨ੍ਹਾਂ ਲਈ ਉਹਨਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ ਅਤੇ ਇਹੀ ਕਾਰਨ ਹੈ ਕਿ ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਯੂਜ਼ਰ ਦੁਆਰਾ ਪਰਿਭਾਸ਼ਿਤ ਨਾਮ (ਆਈਡੈਂਟੀਫਾਇਰਜ਼) ਵਜੋਂ ਨਹੀਂ ਕੀਤੀ ਜਾ ਸਕਦੀ। ਅਸੀਂ ਕੀਵਰਡਜ਼ ਦੇ ਅਰਥ ਨਹੀਂ ਬਦਲ ਸਕਦੇ॥ ਇਹ ਕੀਵਰਡਜ਼ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਜਿੱਥੇ ਵੀ ਲੋੜੀਂਦੇ ਹਨ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਸਾਰੇ ਕੀਵਰਡ ਸਿਰਫ ਛੋਟੇ ਅੱਖਰਾਂ Lower Case ਵਿੱਚ ਲਿਖੇ ਜਾਣੇ ਚਾਹੀਦੇ ਹਨ। ਕਿਉਂਕਿ ਜਾਵਾ ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲ (Case-Sensitive) ਭਾਸ਼ਾ ਹੈ, ਇਸ ਲਈ ਜੇਕਰ ਅਸੀਂ ਇਹਨਾਂ ਕੀਵਰਡਜ਼ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੱਡੇ-ਕੇਸ (upper-case) ਵਿੱਚ ਲਿਖਦੇ ਹਾਂ, ਤਾਂ ਇਹ ਕੰਪਾਇਲੇਸ਼ਨ ਐਰਰਜ਼ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰੇਗਾ। ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲ ਭਾਸ਼ਾ ਇੱਕ ਅਜਿਹੀ ਭਾਸ਼ਾ ਹੁੰਦੀ ਹੈ।

ਜੋ ਛੋਟੇ ਅੱਖਰਾਂ ਅਤੇ ਵੱਡੇ ਅੱਖਰਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਤੱਤਾਂ ਵਜੋਂ ਮੰਨਦੀ ਹੈ॥ਜਾਵਾ ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਵੱਖ-ਵੱਖ ਕੀਵਰਡਜ਼ ਹੇਠਾਂ ਸੂਚੀਬੱਧ ਕੀਤੇ ਗਏ ਹਨ:

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

ਟੇਬਲ 4.1 ਜਾਵਾ ਕੀਵਰਡਜ਼ ਦੀ ਸੂਚੀ

4.8.2.2 ਆਈਡੈਂਟੀਫਾਇਰਜ਼ (Identifiers) : ਆਈਡੈਂਟੀਫਾਇਰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟ (ਤੱਤ) ਜਿਵੇਂ ਕਿ: ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟਸ, ਐਰੇ, ਮੈਥਡਜ, ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ, ਆਦਿ ਨੂੰ ਦਿੱਤੇ ਗਏ ਨਾਮ ਹੁੰਦੇ ਹਨ। ਹਰੇਕ ਪ੍ਰੋਗਰਾਮ ਐਲੀਮੈਂਟ ਨੂੰ ਦੂਜੇ ਐਲੀਮੈਂਟਸ ਤੋਂ ਵੱਖਰਾ ਕਰਨ ਲਈ ਨਾਮ ਦਿੱਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ। ਐਲੀਮੈਂਟਸ ਨੂੰ ਦਿੱਤਾ ਗਿਆ ਨਾਮ ਸਾਰਥਕ (meaningful) ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਤਾਂ ਜੋ ਪ੍ਰੋਗਰਾਮ ਐਲੀਮੈਂਟਸ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਸਮਝਿਆ ਜਾ ਸਕੇ। ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟਸ ਨੂੰ ਨਾਮ ਦੇਣ ਤੋਂ ਬਾਅਦ ਉਹਨਾਂ ਨੂੰ ਉਹਨਾਂ ਦੇ ਨਾਮ ਦੁਆਰਾ ਪਛਾਣਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਐਲੀਮੈਂਟਸ ਦੇ ਨਾਮ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ, ਕੁਝ ਨਾਮਕਰਨ ਨਿਯਮਾਂ (naming rules) ਦੀ ਪਾਲਣਾ ਕੀਤੀ ਜਾਣੀ ਚਾਹੀਦੀ ਹੈ। ਇਹ ਨਾਮਕਰਨ ਨਿਯਮ ਹੇਠਾਂ ਦਿੱਤੇ ਗਏ ਹਨ:

a) ਆਈਡੈਂਟੀਫਾਇਰ ਵਿੱਚ ਵੱਡੇ ਅਤੇ ਛੋਟੇ ਅੱਖਰ (Uppercase and Lowercase Letters), ਅੰਕ (Digits), ਡਾਲਰ ਚਿੰਨ੍ਹ ($) ਅਤੇ ਇੱਕ ਅੰਡਰਸਕੋਰ (_) ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ।
b) ਆਈਡੈਂਟੀਫਾਇਰ ਅੱਖਰ (letter), ਡਾਲਰ ਦੇ ਚਿੰਨ੍ਹ ਜਾਂ ਇੱਕ ਅੰਡਰਸਕੋਰ ਨਾਲ ਸ਼ੁਰੂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ, ਭਾਵ ਇਹ ਅੰਕ ਨਾਲ ਸ਼ੁਰੂ ਨਹੀਂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।
c) ਆਈਡੈਂਟੀਫਾਇਰ ਅੱਖਰ-ਸੰਵੇਦਨਸ਼ੀਲ (Case-Sensitive) ਹੁੰਦੇ ਹਨ, ਭਾਵ ਵੱਡੇ ਅੱਖਰਾਂ (Uppercase) ਵਿੱਚ ਲਿਖਿਆ ਹੋਇਆ ਆਈਡੈਂਟੀਫਾਇਰ ਛੋਟੇ ਅੱਖਰਾਂ (Lowercase) ਵਿੱਚ ਲਿਖੇ ਗਏ ਆਈਡੈਂਟੀਫਾਇਰ ਤੋਂ ਵੱਖਰਾ (different) ਹੁੰਦਾ ਹੈ।
d) ਇੱਕ ਕੀਵਰਡ ਨੂੰ ਆਈਡੈਂਟੀਫਾਇਰ ਵਜੋਂ ਨਹੀਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਕਿਉਂਕਿ ਇਹ ਇੱਕ ਰਾਖਵਾਂ ਸ਼ਬਦ (reserved word) ਹੁੰਦਾ ਹੈ ਅਤੇ ਇਸਦਾ ਇਕ ਵਿਸ਼ੇਸ਼ ਅਰਥ ਹੁੰਦਾ ਹੈ।
e) ਪ੍ਰੋਗਰਾਮ ਜਾਂ ਸਕੋਪ ਦੇ ਇੱਕ ਸੈਕਸ਼ਨ ਅੰਦਰ, ਹਰੇਕ ਯੂਜ਼ਰ-ਪਰਿਭਾਸ਼ਿਤ ਆਈਟਮ (User Defined Item) ਦਾ ਇੱਕ ਵਿਲੱਖਣ (Unique) ਆਈਡੈਂਟੀਫਾਇਰ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।
f) ਆਈਡੈਂਟੀਫਾਇਰ ਕਿਸੇ ਵੀ ਲੰਬਾਈ ਦੇ ਹੋ ਸਕਦੇ ਹਨ।
g) ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਵਿੱਚ ਵਾਈਟ ਸਪੇਸ (white space) ਅਤੇ ਹੋਰ ਅੱਖਰ ਨਹੀਂ ਹੋਣੇ ਚਾਹੀਦੇ ,ਜਿਵੇਂ ਕਿ-*, ; ਆਦਿ। ਹੇਠਾਂ ਸਹੀ **(Legal)** ਅਤੇ ਗਲਤ **(IIIegal)** ਨਾਮ ਵਾਲੇ ਆਈਡੈਂਟੀਫਾਇਰਜ਼ ਦੀਆਂ ਕੁਝ ਉਦਾਹਰਣਾਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਹਨ:

❖ **ਸਹੀ ਨਾਮ ਵਾਲੇ ਆਈਡੈਂਟੀਫਾਇਰਜ਼ (Legal identifiers) ਦੀਆਂ ਉਦਾਹਰਣਾਂ :**

MinNumber, total, vk49, hello_world, $amount, _under_value

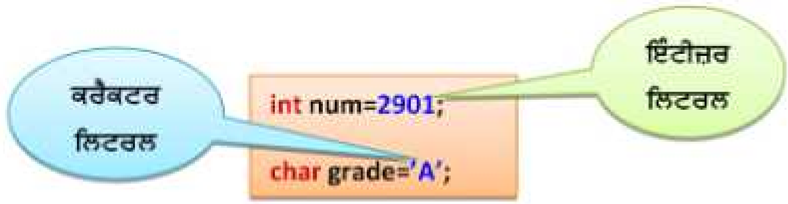
❖ ਗਲਤ ਨਾਮ ਵਾਲੇ ਆਈਡੈਂਟੀਫਾਇਰਜ਼ (illegal identifiers) ਦੀਆਂ ਉਦਾਹਰਣਾਂ :

4avk, amount, Roll No, Hous

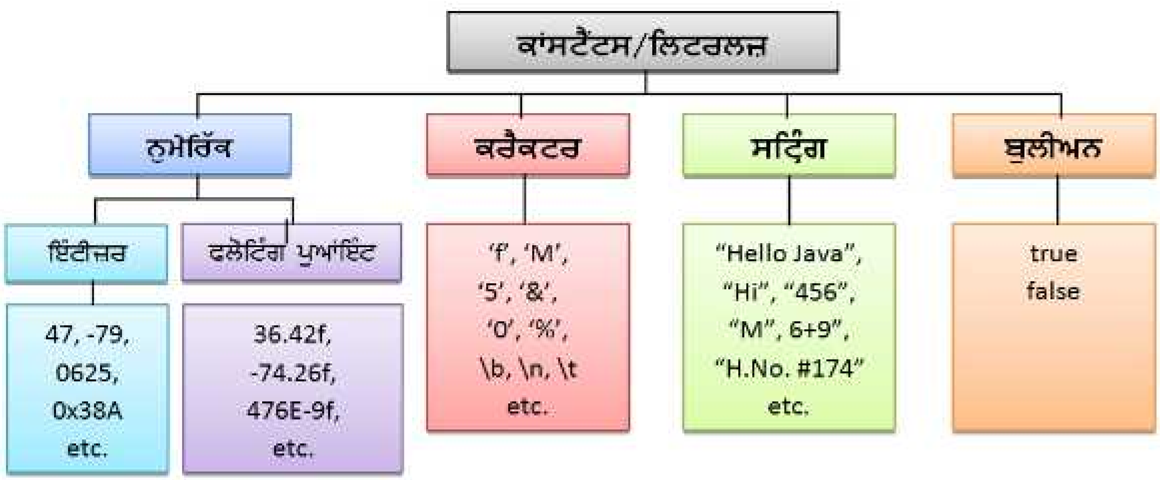
ਜਾਵਾ ਦੇ ਸਟੈਂਡਰਡ ਨੇਮਿੰਗ ਕਨਵੈਨਸ਼ਨਜ਼ ਅਨੁਸਾਰ, ਵੇਰੀਏਬਲ ਲਈ ਸਾਰੇ ਲੈਟਰਜ਼ (letters) ਛੋਟੇ-ਅੱਖਰਾਂ (Lowercase) ਵਿੱਚ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ। ਵੇਰੀਏਬਲਾਂ ਦੇ ਨਾਵਾਂ ਲਈ ਅੰਡਰਸਕੋਰ ਦੀ ਸਿਫ਼ਾਰਸ਼ ਨਹੀਂ ਕੀਤੀ ਜਾਂਦੀ। ਕਾਂਸਟੈਂਟਸ (static, final, attributes ਅਤੇ enums) ਲਈ ਸਾਰੇ ਲੈਟਰਜ਼ (letters) ਵੱਡੇ ਅੱਖਰਾਂ (Uppercase) ਵਿੱਚ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ।

4.8.2.3 ਲਿਟਰਲਜ਼ (Literals):

ਲਿਟਰਲਜ਼ ਨੂੰ ਸਥਿਰ ਮੁੱਲ (Constant Values) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਨਿਸ਼ਚਿਤ (Fixed) ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਆਮ ਤੌਰ 'ਤੇ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੱਤਾਂ ਲਈ ਨਿਰਧਾਰਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ: 2901, 47.29f, "Param", 'A' ਆਦਿ ਸਾਰੇ ਸਥਿਰ ਮੁੱਲ ਹਨ।



ਮੁੱਲ ਦੀ ਕਿਸਮ ਦੇ ਆਧਾਰ 'ਤੇ C ਲਿਟਰਲਕਾਂਸਟੈਂਟਸ ਨੂੰ ਮੋਟੇ ਤੌਰ 'ਤੇ ਚਾਰ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ: ਨੁਮੇਰਿੱਕ (Numeric) ਕਾਂਸਟੈਂਟਸ, ਕਰੈਕਟਰ (Character) ਕਾਂਸਟੈਂਟਸ, ਸਟਿੰਗ (String) ਕਾਂਸਟੈਂਟਸ ਅਤੇ ਬੁਲੀਅਨ (Boolean) ਕਾਂਸਟੈਂਟਸ :



ਚਿੱਤਰ: 4.17 ਜਾਵਾ ਵਿਚ ਲਿਟਰਲ ਕਾਂਸਟੈਂਟਸ ਦੀਆਂ ਕਿਸਮਾਂ

ੳ. ਨੁਮੇਰਿੱਕ ਲਿਟਰਲਜ਼ (Numeric Literals) :

ਨੁਮੇਰਿਕ ਲਿਟਰਲਜ਼ ਉਹ ਸੰਖਿਆਤਮਕ ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਸੰਖਿਆਤਮਕ ਗਣਨਾਵਾਂ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹਨਾਂ ਦਾ ਮੁੱਲ ਅੰਕਾਂ ਦੀ ਲੜੀ (squence of digits) ਦੇ ਰੂਪ ਵਿਚ(ਦਸ਼ਮਲਵ ਬਿੰਦੂ ਦੇ ਨਾਲ ਜਾਂ ਦਸ਼ਮਲਵ ਬਿੰਦੂ ਤੋਂ ਬਿਨਾਂ) ਹੁੰਦਾ ਹੈ ਜੋ ਸਕਾਰਾਤਮਕ (Positive) ਜਾਂ ਨਕਾਰਾਤਮਕ (Negative) ਹੋ ਸਕਦਾ ਹੈ। ਮੂਲ ਰੂਪ ਵਿੱਚ (By default), ਨੁਮੇਰਿੱਕ ਲਿਟਰਲਜ਼ ਨੂੰ ਅੱਗੇ ਦੋ ਕਿਸਮਾਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ:

i. **ਇੰਟੀਜ਼ਰ ਲਿਟਰਲਜ਼ (Integer Literals) :** ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿਚ ਅੰਸ਼ਿਕ/ਦਸ਼ਮਲਵ (fractional/decimal) ਭਾਗ ਨਹੀਂ ਹੁੰਦਾ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਸਕਾਰਾਤਮਕ (+) ਜਾਂ ਨਕਾਰਾਤਮਕ (−) ਚਿੰਨ ਦੇ ਨਾਲ 0 ਤੋਂ 9 ਤੱਕ ਦੇ ਅੰਕ ਹੁੰਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ: 56, +26, −96 ਆਦਿ ਇੰਟੀਜ਼ਰ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ। ਕਿਸੇ ਮੁੱਲ ਨੂੰ ਲਾਂਗ ਇੰਟੀਜ਼ਰ (long integer) ਵਜੋਂ ਦਰਸਾਉਣ ਲਈ ਅਸੀਂ ਉਸ ਮੁੱਲ ਦੇ ਨਾਲ ਪਿਛੇਤਰ (suffix) ਵਜੋਂ L ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ, ਉਦਾਹਰਣ ਵੱਜੋਂ 3456L ਲਿਟਰਲ, ਇਹ ਲਾਂਗ ਇੰਟੀਜਰ ਨੂੰ ਦਰਸਾਵੇਗਾ। ਇੰਟੀਜ਼ਰ ਲਿਟਰਲਜ਼ ਨੂੰ ਤਿੰਨ ਵੱਖ-ਵੱਖ ਨੰਬਰ ਸਿਸਟਮਾਂ ਦੁਆਰਾ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ:

- **ਡੈਸੀਮਲ (Decimal (base 10)) :** ਕਿਸੇ ਸੰਖਿਆ ਨੂੰ ਡੈਸੀਮਲ (ਦਸ਼ਮਲਵ) ਫਾਰਮੈਟ ਵਿਚ ਦਰਸਾਉਣ ਲਈ ਉਸਦੇ ਸਭ ਤੋਂ ਖੱਬੇ (left most) ਅੰਕ ਨੂੰ ਗੈਰ-ਜ਼ੀਰੋ (Non-Zero) ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ: 56, 5689 ਆਦਿ।
- **ਓਕਟਲ (Octal (base 8)) :** ਅਸੀਂ 0 ਤੋਂ 7 ਅੰਕਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਬਣੇ ਨੰਬਰਾਂ ਤੋਂ ਪਹਿਲਾਂ ਜ਼ੀਰੋ ਲਗਾ ਕੇ ਉਸ ਨੰਬਰ ਨੂੰ ਓਕਟਲ ਫਾਰਮੈਟ ਵਿਚ ਦਰਸਾ ਸਕਦੇ ਹਾਂ। ਉਦਾਹਰਣ ਲਈ: 0625, 034 ਆਦਿ।
- **ਹੈਕਸਾਡੈਸੀਮਲ (Hexadecimal (base 16)) :** ਕਿਸੇ ਸੰਖਿਆ ਨੂੰ ਹੈਕਸਾਡੈਸੀਮਲ ਫਾਰਮੈਟ ਵਿਚ ਦਰਸ਼ਾਉਣ ਲਈ ਉਸ ਸੰਖਿਆ ਤੋਂ ਪਹਿਲਾਂ Ox ਜਾਂ OX ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਲਈ: 0×38A, 0×C4 ਆਦਿ।

ii. **ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਲਜ਼ (Floating Point Literals) :** ਇਹਨਾਂ ਨੂੰ ਰੀਅਲ (ਅਸਲ) ਸੰਖਿਆਵਾਂ (Real Numbers) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਫੈਕਸ਼ਨਲ/ਦਸ਼ਮਲਵ (fractional/decimal) ਭਾਗ ਵਾਲੀਆਂ ਸੰਖਿਆਵਾਂ ਹੁੰਦੀਆਂ ਹਨ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਸਕਾਰਾਤਮਕ (+) ਜਾਂ ਨਕਾਰਾਤਮਕ (−) ਚਿੰਨ੍ਹ ਦੇ ਨਾਲ 0 ਤੋਂ 9 ਤੱਕ ਦੇ ਅੰਕ ਹੁੰਦੇ ਹਨ। ਇਸ ਵਿੱਚ ਇੱਕ ਦਸ਼ਮਲਵ ਬਿੰਦੂ (.) ਵੀ ਹੁੰਦਾ ਹੈ ਜੋ ਰੀਅਲ ਸੰਖਿਆ ਦੇ ਇੰਟੀਜ਼ਰ (Integral) ਅਤੇ ਫ੍ਰੈਕਸ਼ਨ (Fractional) ਭਾਗਾਂ ਨੂੰ ਵੱਖ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਇੰਟੀਜ਼ਰ (Integral) ਭਾਗ ਨਾਲ ਕੋਈ ਚਿੰਨ੍ਹ ਨਹੀਂ ਹੈ, ਤਾਂ ਇਸਨੂੰ ਸਕਾਰਾਤਮਕ (Postive) ਰੀਅਲ ਲਿਟਲ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: 3.14, +256.5896, 96.14, 36.00 ਆਦਿ ਰੀਅਲ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ। ਜਾਵਾ ਵਿੱਚ ਦੋ ਤਰ੍ਹਾਂ ਦੇ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਹੁੰਦੇ ਹਨ: ਫਲੋਟ (float) ਅਤੇ ਡਬਲ (double) ਅਸੀਂ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਲਿਟਰਲਜ਼ ਲਈ ਇੱਕ ਪਿਛੇਤਰ ਜਿਵੇਂ ਕਿ D, d, F, f ਵੀ ਲਗਾ ਸਕਦੇ ਹਾਂ। ਪਿਛੇਤਰ ਦ ਜਾਂ ਧ ਦੀ ਵਰਤੋਂ ਡਬਲ-ਪੀਸੀਜ਼ਨ (Double-Precision) ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਲ ਮੰਨਿਆਂ ਜਾਂਦਾ ਹੈ।ਉਦਾਹਰਣ ਲਈ:

65563.554	Double-precision floating-point literal (by default)
34578.343D	Double-precision floating-point literal
45344.01f	Floating-point literal

ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਜ਼ ਨੂੰ ਤਰੀਕਿਆਂ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ ਅਤੇ ਵਿਗਿਆਨਕ ਨੋਟਸ਼ਨ 1.

- **ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ (Standard Notation) :** ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ ਵਿੱਚ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਇੱਕ ਪੂਰਨ ਅੰਕ (integer) ਅਤੇ ਇੱਕ ਫਰੈਕਸ਼ਨ (fractional) ਹਿੱਸੇ ਤੋਂ ਬਣਿਆ ਹੁੰਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਦੋਵੇਂ ਹਿੱਸਿਆਂ ਵਿਚਕਾਰ ਇੱਕ ਦਸ਼ਮਲਵ ਬਿੰਦੂ (decimal point) ਲੱਗਿਆ ਹੁੰਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: 130.35, 41.0, 0.56, 0.67, 0.67, 7.71 ਆਦਿ।
- **ਵਿਗਿਆਨਕ ਨੋਟੇਸ਼ਨ (Scientific Notation) :** ਵਿਗਿਆਨਕ ਨੋਟੇਸ਼ਨ ਵਿੱਚ ਇੱਕ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਲ ਵਿੱਚ ਇੱਕ ਮੈਂਟੀਸਾ (mantissa) ਅਤੇ ਇੱਕ ਐਕਸਪੋਨੈਂਟ (exponent) ਭਾਗ ਹੁੰਦਾ ਹੈ। ਮੈਨਟੀਸਾ ਭਾਗ ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ ਵਿੱਚ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਅਤੇ ਐਕਸਪੋਨੈਂਟ ਭਾਗ 10 ਦੀ ਉਸ ਪਾਵਰ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ ਜਿਸ ਨਾਲ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਨੂੰ ਗੁਣਾ ਕੀਤਾ ਜਾਣਾ ਹੈ। ਮੈਨਟੀਸਾ ਭਾਗ ਅਤੇ ਐਕਸਪੋਨੈਂਟ ਭਾਗ ਨੂੰ ਅੱਖਰ **E** (ਅਪਰਕੇਸ ਜਾਂ ਲੋਅਰਕੇਸ) ਦੁਆਰਾ ਵੱਖ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਇੱਕ ਨੰਬਰ 231.54 ਨੂੰ 2.3154e2 (2.3154E102) ਨਾਲ ਐਕਸਪੋਨੈਂਟ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇੱਥੇ 2.3154 ਮੈਨਟੀਸਾ ਭਾਗ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਅਤੇ ਅੱਖਰ ਦਾ **e** ਤੋਂ ਬਾਅਦ ਵਾਲਾ ਹਿੱਸਾ 2 ਐਕਸਪੋਨੈਂਟ ਭਾਗ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜਿਸਦਾ ਅਧਾਰ ਮੁੱਲ 10 ਹੁੰਦਾ ਹੈ। ਐਕਸਪੋਨੈਂਟ ਰੂਪ

ਬਹੁਤ ਵੱਡੀਆਂ ਸੰਖਿਆਵਾਂ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਉਪਯੋਗੀ ਹੁੰਦਾ ਹੈ ਕਿਉਂਕਿ ਜਦੋਂ ਕਿਸੇ ਵੱਡੀ ਸੰਖਿਆ ਨੂੰ ਇਸ ਰੂਪ ਵਿਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਉਸ ਸੰਖਿਆ ਨਾਲ ਲਿਖੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਜ਼ੀਰੋਆਂ ਦੀ ਵੱਡੀ ਗਿਣਤੀ ਨੂੰ ਲਿਖਣ ਤੋਂ ਬਚਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: 6000000 ਨੂੰ 6.0e6 ਵਜੋਂ ਲਿਖਿਆ ਜਾ ਸਕਦਾ ਹੈ।

ਅ. ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ (Character Literals) : ਇਹ ਸਿੰਗਲ ਕਰੈਕਟਰ ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਆਮ ਤੌਰ 'ਤੇ ਗਣਨਾਵਾਂ ਵਿੱਚ ਸ਼ਾਮਲ ਨਹੀਂ ਹੁੰਦੇ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਨੂੰ ਸਿੰਗਲ ਕੋਮਿਆ (single quotes) ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਇੱਕ ਤੋਂ ਵੱਧ ਪ੍ਰਿੰਟੇਬਲ ਕਰੈਕਟਰਾਂ ਨੂੰ ਰੱਖਣ ਦੀ ਆਗਿਆ ਨਹੀਂ ਹੁੰਦੀ।ਨਾਨ-ਪ੍ਰਿੰਟੇਬਲ ਕਰੈਕਟਰ ਵੀ ਇਸ ਸ਼੍ਰੇਣੀ ਵਿੱਚ ਆਉਂਦੇ ਹਨ ਹਾਲਾਂਕਿ ਇਨ੍ਹਾਂ ਕਰੈਕਟਰਾਂ ਵਿੱਚ ਦੋ ਚਿੰਨ ਵਰਤੇ ਜਾਂਦੇ ਹਨ, ਉਦਾਹਰਣ ਵਜੋਂ ਨਵਾਂ ਲਾਈਨ ਅੱਖਰ (\n - ਬੈਕਸਲੈਸ਼ (/) ਅਤੇ ਇੱਕ ਅੱਖਰ n), ਪਰ ਫਿਰ ਵੀ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਕਰੈਕਟਰ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕਹਿਰੇ ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਕੁੱਝ ਉਦਾਹਰਣ ਇਸ ਪ੍ਰਕਾਰ ਹਨ: 'A', 'g', '', +, $, \n, \t, ਆਦਿ। ਮੁੱਲ 'ab', '45' ਆਦਿ ਇੱਕਹਿਰੇ ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਸਹੀ ਉਦਾਹਰਣਾਂ ਨਹੀਂ ਹਨ ਕਿਉਂਕਿ ਇਹਨਾਂ ਵਿੱਚ ਇੱਕ ਤੋਂ ਵੱਧ ਅੱਖਰਾਂ ਨੂੰ ਸਿੰਗਲ ਕਾਮਿਆਂ ਵਿੱਚ ਰੱਖਿਆ ਗਿਆ ਹੈ ਜਿਸ ਦੀ ਇੱਕਹਿਰੇ ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਆਗਿਆ ਨਹੀਂ ਹੁੰਦੀ।

ੲ. ਸਟਿੰਗ ਲਿਟਰਲਜ਼ (String Literals) : ਇਹ ਲਿਟਰਲਜ਼ ਕਿਸੇ ਵੀ ਗਿਣਤੀ ਵਿਚ ਕਰੈਕਟਰਾਂ ਦੇ ਸਮੂਹ(combination of any number of characters) ਨੂੰ ਦੋਹਰੇ ਕਾਮਿਆਂ (double quotes) ਵਿੱਚ ਰੱਖ ਕੇ ਲਿਖੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹ ਲਿਟਰਲਜ਼ ਅੰਗਰੇਜ਼ੀ ਦੇ ਅੱਖਰਾਂ, ਅੰਕਾਂ, ਖਾਸ ਚਿੰਨ੍ਹਾਂ ਅਤੇ ਖਾਲੀ ਥਾਵਾਂ ਆਦਿ ਦੇ ਸਮੂਹ ਨਾਲ ਬਣੇ ਹੋ ਸਕਦੇ ਹਨ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਕੁੱਝ ਉਦਾਹਰਣਾਂ ਇਸ ਪ੍ਰਕਾਰ ਹਨ: "Paramveer", "A", "House#196", "1829" ਆਦਿ।

ਸ. ਬੂਲੀਅਨ ਲਿਟਰਲਜ਼ (Boolen Literals) : ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ true ਅਤੇ false ਮੁੱਲਾਂ ਨੂੰ ਵੀ ਲਿਟਲ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਬੂਲੀਅਨ ਟਾਈਪ ਦੇ ਵੇਰੀਏਬਲ ਲਈ ਕੋਈ ਮੁੱਲ ਸੈੱਟ ਕਰਨਾ ਹੋਵੇ ਤਾਂ ਅਸੀਂ ਸਿਰਫ ਇਹਨਾਂ ਦੋ ਮੁੱਲਾਂ ਦੀ ਹੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ। C ਭਾਸ਼ਾ ਤੋਂ ਉਲਟ ਅਸੀਂ ਜਾਵਾ ਵਿਚ ਇਹ ਨਹੀਂ ਮੰਨ ਸਕਦੇ ਕਿ 1 ਦਾ ਮੁੱਲ true ਦੇ ਬਰਾਬਰ ਹੈ ਅਤੇ 0 false ਦੇ ਬਰਾਬਰ ਹੈ। ਬੁਲੀਅਨ ਮੁੱਲ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਸਾਨੂੰ ture ਅਤੇ false ਮੁੱਲਾਂ ਦੀ ਹੀ ਵਰਤੋਂ ਕਰਨੀ ਪਵੇਗੀ। ਉਦਾਹਰਣ ਲਈ:

```
Boolean flag = true;
```

4.8.2.4 ਆਪਰੇਟਰਜ਼ (Operators) :

ਆਪਰੇਟਰ ਉਹ ਚਿੰਨ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੁੱਝ ਗਣਿਤਕ ਜਾਂ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ॥ ਉਦਾਹਰਣ ਲਈ: +,-,#, %, ++, -- ਆਦਿ। ਆਪਰੇਟਰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਮੁੱਲਾਂ ਵੇਰੀਏਬਲਾਂ ਉੱਪਰ ਕੰਮ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਇਹਨਾਂ ਮੁੱਲਾਂ/ਵੇਰੀਏਬਲਾਂ ਨੂੰ **ਓਪਰੈਂਡਜ਼** (operands) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਸੀ ਭਾਸ਼ਾ ਵਿੱਚ ਬਹੁਤ ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਪਹਿਲਾਂ ਤੋਂ ਹੀ ਬਣੇ ਹੋਏ ਹਨ। ਇਹਨਾਂ ਸਾਰੇ ਓਪਰੇਟਰਾਂ ਨੂੰ ਤਿੰਨ ਵਿਆਪਕ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ: ਯੂਨਰੀ (unary), ਬਾਈਨਰੀ (binary) ਅਤੇ ਟਰਨਰੀ (Ternary). ਜਾਵਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਓਪਰੇਟਰਾਂ ਦੀ ਵਿਸਤਾਰ ਵਿਚ ਵਿਆਖਿਆ ਅਗਲੇ ਪਾਠ ਵਿਚ ਕੀਤੀ ਗਈ ਹੈ।

4.8.2.5 ਪੰਕਚੁਏਟਰਜ਼ (Punctuators):

ਇਹਨਾਂ ਚਿੰਨ੍ਹਾਂ ਨੂੰ ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਹਰੇਕ ਚਿੰਨ੍ਹਾਂ ਦੀ ਆਪਣੀ ਵੱਖਰੀ ਵਿਸ਼ੇਸ਼ਤਾ ਹੁੰਦੀ ਹੈ ॥ਹਰ ਇੱਕ ਚਿੰਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੁੱਝ ਖਾਸ ਕੰਮ ਦਰਸਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਦੇ ਤੌਰ ਤੇ: ਸੈਮੀਕਾਲਨ; ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਸਟੇਟਮੈਂਟ ਨੂੰ ਖਤਮ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਕਾਮੇ (,) ਨੂੰ ਸੈਪਰੇਟਰ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ, ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਗੋਲ ਬਰੈਕਟ (()), ਐਰੇ ਲਈ ਚਕੋਰ ਬਰੈਕਟ ([]) ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਗਰੁੱਪ ਕਰਨ ਲਈ ਘੁੰਡੀਦਾਰ ਬਰੈਕਟ ({}) ਵਰਤੇ ਜਾਂਦੇ ਹਨ ॥

4.8.3 ਕਮੈਂਟਸ (Comments) :

ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਸਿਰਫ਼ ਡਾਕੂਮੈਂਟੇਸ਼ਨ (documentation) ਦੇ ਮੰਤਵ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ॥ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਅੰਦਰ ਕੋਡ ਦਾ ਵਰਣਨ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜੇਕਰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਤਾਂ ਕੋਡ ਨੂੰ ਸਮਝਣਾ ਆਸਾਨ ਹੋ ਜਾਂਦਾ ਹੈ। ਕੰਪਾਈਲਰ ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ ਕਮੈਂਟਸ ਨੂੰ ਨਜ਼ਰਅੰਦਾਜ਼ (ignores) ਕਰਦਾ ਹੈ। ਕਮੈਂਟਸ ਲਈ ਜਾਵਾ ਵਿੱਚ ਹੇਠ ਲਿਖੇ ਸਟਾਈਲਜ਼ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ:

- **ਬਲਾਕ ਸਟਾਈਲ ਕਮੈਂਟਸ (Block style comments)** ਦੀ ਵਰਤੋਂ ਕਈ ਲਾਈਨਾਂ ਵਾਲੇ ਕਮੈਂਟਸ (Multiple Line comments) ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਕਮੈਂਟਸ /* ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦੇ ਹਨ ਅਤੇ */ ਨਾਲ ਖਤਮ ਹੁੰਦੇ ਹਨ।
- **ਲਾਈਨ ਸਟਾਈਲ ਕਮੈਂਟਸ (Line style comments)** ਦੀ ਵਰਤੋਂ ਇਕ ਸਿੰਗਲ ਲਾਈਨ ਕਮੈਂਟ (single line comment) ਬਨਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ।ਇਹ ਕਮੈਂਟ // ਚਿੰਨ੍ਹ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਆਬਜੈਕਟਸ ਅਸਲ ਸੰਸਾਰ ਦੀਆਂ ਵਸਤੂਆਂ (entities) ਹੁੰਦੀਆਂ ਹਨ ਜਿਵੇਂ ਕਿ: ਵਿਦਿਆਰਥੀ, ਵਿਅਕਤੀ, ਪੈਨ, ਟੇਬਲ, ਟੈਲੀਵਿਜ਼ਨ, ਕੰਪਿਊਟਰ ਆਦਿ।
2. ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਜਿੱਥੇ ਹਰ ਚੀਜ਼ ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਟਰੂਲੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ *(Truly Object-Oriented Progamming Languge)* ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
3. OOP ਵਿੱਚ ਕੋਡ ਅਤੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਇਕਾਈ - ਆਬਜੈਕਟ (object) ਵਿਚ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
4. ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਦੇ ਚਾਰ ਮੁੱਖ ਸਿਧਾਂਤ ਹੁੰਦੇ ਹਨ: ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction), ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਇਨਹੈਰੀਟੈਂਸ (Interitance) ਅਤੇ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism)
5. ਇੱਕ ਕਲਾਸ ਉਹਨਾਂ ਆਬਜੈਕਟਸ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦੀ ਹੈ, ਜਿਹਨਾਂ ਵਿੱਚ ਇਕ ਸਮਾਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (same properties) ਅਤੇ ਸਾਂਝਾ ਵਿਵਹਾਰ (common behaviour) ਹੁੰਦਾ ਹੈ॥
6. ਐਬਸਟਰੈਕਸ਼ਨ ਸਿਧਾਂਤ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਂਟੀਟੀ ਵਿੱਚ ਇੱਕ ਅਸਲ ਸੰਸਾਰ ਐਂਟੀਟੀ (real world entity) ਨੂੰ ਕਿਵੇਂ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।
7. ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਸਮੇਟਣ (wrapping) ਦੀ ਇੱਕ ਪ੍ਰਕਿਰਿਆ ਹੈ। ਇਸ ਸਿੰਗਲ ਯੂਨਿਟ ਨੂੰ ਕਲਾਸ (class) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
8. ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਸਾਡੀ ਜ਼ਰੂਰਤ ਅਨੁਸਾਰ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਛੁਪਾ ਕੇ (selective hiding of properties and methods) ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਲਪੇਟ (wrapping) ਕੇ ਰੱਖਦੀ ਹੈ, ਜਿਸਨੂੰ ਕਲਾਸ (class) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
9. ਇਨਹੈਰੀਟੈਂਸ ਇੱਕ ਅਜਿਹੀ ਵਿਧੀ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੇ ਪੇਰੈਂਟ (parent) ਆਬਜੈਕਟ ਦੀਆਂ ਸਾਰੀਆਂ ਅਵਸਥਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰਾਂ (states and behaviours) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।
10. ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਕਈ ਰੂਪਾਂ ਨੂੰ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
11. ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਦੋ ਕਿਸਮ ਦੇ ਪੋਲੀਮੋਫਿਜ਼ਮ ਹਨ: ਕੰਪਾਈਲ-ਟਾਈਮ ਪੋਲੀਮੋਫਿਜ਼ਮ ਅਤੇ ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਫਿਜ਼ਮ।
12. ਕੰਪਾਈਲ ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overriding) ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਸਿਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
13. ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overriding) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
14. Java ਨੂੰ ਸਾਲ 1995 ਵਿੱਚ ਸਨ ਮਾਇਕ੍ਰੋਸਿਸਟਮਜ਼ (Sun Microsystems) ਕੰਪਨੀ ਵਿੱਚ ਜੇਮਸ ਗੋਸਲਿੰਗ (James Gosling) ਦੁਆਰਾ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਸੀ।
15. ਜਦੋਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਪਲੇਟਫਾਰਮ ਮਸ਼ੀਨ (platform specific machine) ਵਿੱਚ ਕੰਪਾਇਲ ਨਹੀਂ ਹੁੰਦਾ। ਇਸ ਦੀ ਬਜਾਏ ਇਸ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਬਾਈਟ-ਕੋਡ (platform independent bytecode) ਵਿੱਚ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

16. JDK JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਮੁੱਖ ਪਲੇਟਫਾਰਮ ਕੰਪੋਨੈਂਟ ਹੈ।

17. JRE ਵਿੱਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਜਾਵਾ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।

18. JVM ਇੱਕ ਐਬਸਟਰੈਕਟ (abstract) ਮਸ਼ੀਨ ਹੁੰਦੀ ਹੈ ॥ਇਹ ਇੱਕ ਸਪੇਸੀਫੀਕੇਸ਼ਨ (specification) ਹੈ।ਜੋ ਇਕ ਅਜਿਹਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ (runtime environment) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ।ਜਿਸ ਵਿੱਚ ਜਾਵਾ ਬਾਈਟਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

19. ਜਾਵਾ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ WORA (Write Once Run Anywhere) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

20. ਜਾਵਾ ਭਾਸ਼ਾ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਹੈ।ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜਾਵਾ ਦਾ ਪ੍ਰੋਗਰਾਮ ਇੱਕ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਤੋਂ ਦੂਜੀ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਉਪਰ ਆਸਾਨੀ ਨਾਲ ਪ੍ਰਯੋਗ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

21. ਟੋਕਨ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਭ ਤੋਂ ਛੋਟੀਆਂ ਵਿਅਕਤੀਗਤ ਇਕਾਈਆਂ (smallest individual units) ਹੁੰਦੀਆਂ ਹਨ।

22. ਕੀਵਰਡਜ਼ ਨੂੰ ਰਿਜ਼ਰਵ ਵਰਡਜ਼ (Reserve Words) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸ਼ਬਦ ਜਾਵਾ ਕੰਪਾਈਲਰ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਪਰਿਭਾਸ਼ਿਤ ਹੁੰਦੇ ਹਨ।

23. ਆਈਡੈਂਟੀਫਾਇਰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟ ਤੱਤ ਜਿਵੇਂ ਕਿ: ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟਸ, ਐਰੇ, ਮੈਥਡਜ਼, ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ, ਆਦਿ ਨੂੰ ਦਿੱਤੇ ਗਏ ਨਾਮ ਹੁੰਦੇ ਹਨ।

24. ਲਿਟਰਲਜ਼ ਨੂੰ ਸਥਿਰ ਮੁੱਲ (Constant Values) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।ਇਹ ਨਿਸ਼ਚਿਤ (Fixed) ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਆਮ ਤੌਰ ਤੇ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੱਤਾਂ ਲਈ ਨਿਰਧਾਰਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

25. ਆਪਰੇਟਰ ਉਹ ਚਿੰਨ੍ਹ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੁੱਝ ਗਣਿਤਕ ਜਾਂ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

26. ਕਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਅੰਦਰ ਕੋਡ ਦਾ ਵਰਣਨ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਕੰਪਾਈਲਰ ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ ਕਮੈਂਟਸ ਨੂੰ ਨਜ਼ਰਅੰਦਾਜ਼ (ignores) ਕਰਦਾ ਹੈ।

ਅਭਿਆਸ

ਪ੍ਰਸ਼ਨ 1 ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ:

i. OOP ਵਿੱਚ ਕੋਡ ਅਤੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਇਕਾਈ................ ਵਿਚ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਪ੍ਰੋਗਰਾਮ (Program) ਅ. ਕਲਾਸ (class)

ੲ. ਆਬਜੈਕਟ (object) ਸ. ਯੂਨਿਟ (Unit)

ii. ....ਸਿਧਾਂਤ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਕਿ ਇਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਂਟੀਟੀ ਵਿੱਚ ਇੱਕ ਅਸਲ ਸੰਸਾਰ ਐਂਟੀਟੀ (real world entity) ਨੂੰ ਕਿਵੇਂ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

ੳ. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਅ. ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation)

ੲ. ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਸ. ਇਹੇਰੀਟੈਂਸ (Inheritance)

iii. ....ਇੱਕ ਅਜਿਹੀ ਵਿਧੀ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੇ ਪੇਰੈਂਟ (parent) ਆਬਜੈਕਟ ਦੀਆਂ ਸਾਰੀਆਂ ਅਵਸਥਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰਾਂ (states and behaviours) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।

ੳ. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਅ. ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation)

ੲ. ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਸ. ਇਨਹੇਰੀਟੈਂਸ (Inheritance)

iv. .... ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading) ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਸਿਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਰਨ-ਟਾਈਮ (Run-Time) ਅ. ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਟਾਈਮ (Execution Time)

ੲ. ਕੰਪਾਈਲ ਟਾਈਮ (Compile-Time) ਸ. ਲੋਡਿੰਗ ਟਾਈਮ (Loading independent)

v. ਜਦੋਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ (platfrom independent) .... ਵਿੱਚ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਸੋਰਸ ਕੋਡ (Source Code) ਅ. ਆਬਜੈਕਟ ਕੋਡ (Object Code)

ੲ. ਬਿੱਟ ਕੋਡ (Bit Code) ਸ. ਬਾਈਟ-ਕੋਡ (Byte Code)

vi. .... ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਭ ਤੋਂ ਛੋਟੀਆਂ ਵਿਅਕਤੀਗਤ ਇਕਾਈਆਂ (smallest individual units) ਹੁੰਦੀਆਂ ਹਨ।

ੳ. ਆਬਜੈਕਟਸ (Objects) ਅ. ਟੋਕਨਜ਼ (Tokens)

ੲ. ਲਿਟਰਲਜ਼ (Literals) ਸ. ਕਲਾਸਾਂ (Classes)

vii. ....ਉਹ ਚਿੰਨ੍ਹ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੁੱਝ ਗਣਿਤਕ ਜਾਂ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

ੳ. ਅੰਕ (Digits) ਅ. ਓਪਰੈਂਡਜ਼ (Operands)

ੲ. ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Expressions) ਸ. ਆਪਰੇਟਰਜ਼ (Operators)

2. ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ:

i. .... ਸਾਡੀ ਜ਼ਰੂਰਤ ਅਨੁਸਾਰ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਛੁਪਾ ਕੇ (selective hiding of properties and methods) ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਲਪੇਟ (wrapping) ਕੇ ਰੱਖਦੀ ਹੈ, ਜਿਸਨੂੰ ਕਲਾਸ (class) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ii. ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਕਈ ਰੂਪਾਂ ਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

iii. JAVA ਨੂੰ ਸਾਲ 1995 ਵਿੱਚ ਸਨ ਮਾਇਕ੍ਰੋਸਿਸਟਮਜ਼ (Sun Microsystems) ਕੰਪਨੀ ਵਿੱਚ .... ਦੁਆਰਾ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਸੀ।

iv. .... ਵਿੱਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਜਾਵਾ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।

v. .... ਇੱਕ ਐਬਸਟਰੈਕਟ (abstract) ਮਸ਼ੀਨ ਹੁੰਦੀ ਹੈ। ਇਹ ਇੱਕ ਸਪੈਸੀਫੀਕੇਸ਼ਨ (specification) ਹੈ ਜੋ ਇਕ ਅਜਿਹਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ (runtime environment) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਜਾਵਾ ਬਾਈਟਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

vi. ....ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟ (ਤੱਤ) ਜਿਵੇਂ ਕਿ: ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟਸ, ਐਰੇ, ਮੈਥਡਜ਼, ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ, ਆਦਿ ਨੂੰ ਦਿੱਤੇ ਗਏ ਨਾਮ ਹੁੰਦੇ ਹਨ।

vii. ਕੰਪਾਈਲਰ ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ........................ ਨੂੰ ਨਜ਼ਰਅੰਦਾਜ਼ (ingores) ਕਰਦਾ ਹੈ।

3. ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਆਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (Object-Oriented Programming Paradigm) ਦੇ ਚਾਰ ਮੁੱਖ ਸਿਧਾਂਤ ਕਿਹੜੇ ਹਨ ?

ii. ਆਬਜੈਕਟ (Object) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

iii. ਇਨਹੈਰੀਟੈਂਸ (Inheritance) ਕੀ ਹੈ ?

iv. ਇਨਕੈਪਸੁਲੇਸ਼ਨ (Encapsultion) ਕੀ ਹੈ ?

v. ਤੁਸੀਂ ਜਾਵਾ ਬਾਰੇ ਕੀ ਜਾਣਦੇ ਹੋ ?

vi. JVM ਕੀ ਹੈ ?

vii. ਕੀਵਰਡਜ਼ (Keywords) ਕੀ ਹੁੰਦੇ ਹਨ ?

viii. ਕਮੈਂਟਸ (Comments) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

ix. ਆਪਰੇਟਰਜ਼ (Operators) ਕੀ ਹਨ ?

x. ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਪਲੇਟਫਾਰਮ-ਸੁਤੰਤਰ (Platfrom Independence) ਵਜੋਂ ਕਿਉਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ ?

4 ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਕੀ ਹੈ ? ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਦੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਨਾਂ ਲਿਖੋ।

ii. ਜਾਵਾ ਦੀਆਂ ਕੋਈ ਪੰਜ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦਾ ਵਰਨਣ ਕਰੋ।

iii. ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਮੁੱਢਲੀ ਬਣਤਰ ਲਿਖੋ।।

iv. ਟੋਕਨਜ਼ (Tokens) ਕੀ ਹੁੰਦੇ ਹਨ ? ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਟੋਕਨਜ਼ ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਦਿਓ।

v. ਆਈਡੈਂਟੀਫਾਇਰ (Identifier) ਕੀ ਹੁੰਦੇ ਹਨ ? ਆਈਡੈਂਟੀਫਾਇਰਜ਼ ਦੇ ਨਾਮਕਰਨ ਸੰਬੰਧੀ ਨਿਯਮਾਂ (naming rules) ਦਾ ਵਰਨਣ ਕਰੋ ॥

vi. ਲਿਟਰਲਜ਼ (Literals) ਕੀ ਹੁੰਦੇ ਹਨ ? ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਲਿਟਰਲਜ਼ ਦਾ ਵਰਨਣ ਕਰੋ।