ਜਾਵਾ ਵਿਚ ਡਾਟਾ ਟਾਈਪਸ ਅਤੇ ਆਪਰੇਟਰਜ਼
(Data Types and Operators in Java)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

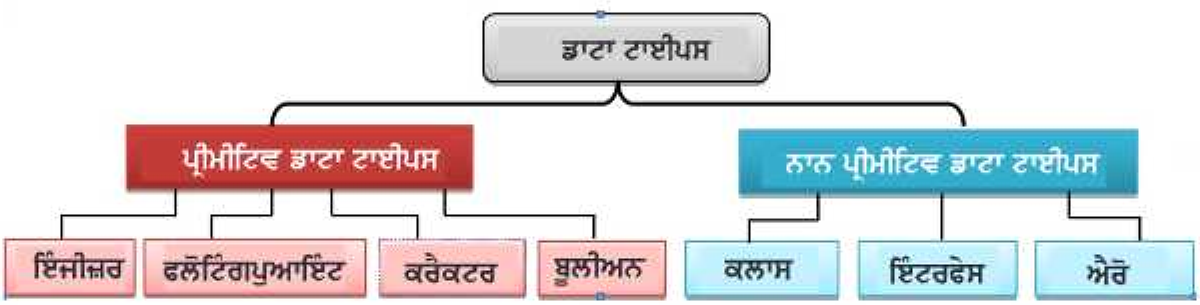
5.1 ਡਾਟਾ ਟਾਈਪਸ (Data Types)
5.2 ਵੇਰੀਏਬਲਜ਼ (Variables)
5.3 ਓਪਰੇਟਰਜ਼ (Operators)
5.4 ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Expressions)
5.5 ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators)
5.6 ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion)

5.1 ਡਾਟਾ ਟਾਈਪਸ (DATA TYPES)

ਡਾਟਾ ਟਾਈਪਸ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦੀਆਂ ਹਨ ਕਿ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੱਤਾਂ ਵਿੱਚ ਕਿਸ ਕਿਸਮ ਦਾ ਡਾਟਾ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ, ਜਿਵੇਂ ਕਿ ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟ, ਐਰੇ ਆਦਿ। ਡਾਟਾ ਟਾਈਪਸ ਇੱਕ ਖਾਸ ਕਿਸਮ, ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ ਅਤੇ ਡਾਟਾ ਉਪਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ। ਜਾਵਾ ਇੱਕ ਸਟ੍ਰਾਂਗਲੀ ਟਾਈਪਡ ਭਾਸ਼ਾ (strongly typed language) ਹੈ ਇਸ ਲਈ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਡਿਕਲੇਰੇਸ਼ਨ (declaration) ਸਮੇਂ ਸਾਰੇ ਵੇਰੀਏਬਲਾਂ ਦੀਆਂ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਡਿਕਲੇਅਰ (ਘੋਸ਼ਿਤ) ਕਰਨ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ।

ਜਾਵਾ ਕਈ ਕਿਸਮਾਂ ਦੀਆਂ ਡਾਟਾ ਟਾਈਪਸ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਅਤੇ ਹਰੇਕ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਕੰਪਿਊਟਰ ਦੀ ਮੈਮੋਰੀ ਵਿੱਚ ਵੱਖਰੇ ਢੰਗ ਨਾਲ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੀਆਂ ਗਈਆਂ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਮੁੱਖ ਤੌਰ 'ਤੇ ਦੋ ਕਿਸਮਾਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

- ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Primitive Data Types)
- ਨਾਨ- ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Non-Primitive Data Types)



ਚਿੱਤਰ 5.1: ਜਾਵਾ ਵਿੱਚ ਡਾਟਾ ਟਾਈਪਸ ਦਾ ਵਰਗੀਕਰਨ

5.1.1 ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Primitive Data Types) : ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਬਿਲਟ-ਇਨ (built-in) ਡਾਟਾ ਟਾਈਪ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਹਨ :

1. ਇੰਟੀਜ਼ਰ ***(Integer):*** byte, short, int, ਅਤੇ long

II. ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ (***Floating Point:*** float ਅਤੇ double)

III. ਕਰੈਕਟਰ (***Character):*** char

IV. ਬੁਲੀਅਨ (***Boolean):*** boolean

ਇਹਨਾਂ ਡਾਟਾ ਟਾਈਪਸ ਦਾ ਵਿਸਥਾਰ ਵਿੱਚ ਵਰਣਨ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

I. ਇੰਟੀਜ਼ਰ ਟਾਈਪ (Integer Type): ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਪੂਰਨ ਅੰਕ ਮੁੱਲਾਂ (ਬਿਨ੍ਹਾਂ ਫ੍ਰੈਕਸ਼ਨਲ (fractional) ਭਾਗ ਵਾਲੇ ਅੰਕ) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਿਵੇਂ ਕਿ:49, 174, +2901, 54896 ਆਦਿ।JAVA ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ: byte, short int. ਅਤੇ long ਟਾਈਪਸ। ਇਹਨਾਂ ਸਾਰੀਆਂ ਡਾਟਾ ਟਾਈਪਸ ਨਾਲ ਸਕਾਰਾਤਮਕ ਜਾਂ ਨਕਾਰਾਤਮਕ ਚਿੰਨ੍ਹ (positive or negative sign) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਅਨਸਾਈਂਡ ਇੰਟੀਜ਼ਰ (unsigned integer) ਦੀ ਕੋਈ ਧਾਰਨਾ ਨਹੀਂ ਹੈ। ਇਹਨਾਂ ਡਾਟਾ ਟਾਈਪਸ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ 0 ਹੁੰਦਾ ਹੈ। ਇਹਨਾਂ ਟਾਈਪਸ ਦੀ ਸਟੋਰੇਜ ਸਮਰੱਥਾ ਵੱਖ-ਵੱਖ ਹੁੰਦੀ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪਸ ਲਈ ਲੋੜੀਂਦੀਆਂ ਮੈਮੋਰੀ ਜ਼ਰੂਰਤਾਂ (memory requirements) ਅਤੇ ਉਹਨਾਂ ਵਿਚ ਸਟੋਰੇ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ਼ ਨੂੰ ਦਰਸਾ ਰਿਹਾ ਹੈ:

ਡਾਟਾ ਟਾਈਪ	ਸਟੋਰੇਜ਼ ਸਾਈਜ਼	ਡਾਟਾ ਦੀ ਰੇਂਜ਼
byte	1 byte	-2^7 to $+2^7 - 1$ OR (-127 to +128)
short	2 bytes	-2^{15} to $+2^{15} - 1$ OR (-32, 768 to +32, 767)
int	4 bytes	-2^{31} to $+2^{31} - 1$ OR (-2, 147, 483, 648 to (+2, 147, 483, 647)
long	8 bytes	-2^{63} to $+2^{63} - 1$ OR (-9, 223, 372, 036, 854, 775, 808 to +9, 223, 372, 036, 854, 775, 808)

ਟੇਬਲ 5.1 ਇੰਟੀਜਰ ਡਾਟਾ ਟਾਇਪ ਦਾ ਸੰਖੇਪ ਵਿੱਚ ਜਾਣਕਾਰੀ

ਅਸੀਂ ਆਪਣੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਪੂਰਨਅੰਕ ਮੁੱਲ (Integer value) ਦੀ ਕਿਸਮ ਅਤੇ ਰੇਂਜ ਦੇ ਅਧਾਰ ਤੇ ਵੇਰੀਏਬਲ ਲਈ ਢੁਕਵੀਂ ਡਾਟਾ ਟਾਈਪ ਦੀ ਚੋਣ ਕਰ ਸਕਦੇ ਹਾਂ॥

II. ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਟਾਈਪ (Floating-Point Type) : ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਡਾਟਾਟਾਈਪ ਦੀ ਵਰਤੋਂ ਰੀਅਲ ਨੰਬਰਾਂ ਜਿਵੇਂ ਕਿ: 3.14, 74,5884569, +29.05, -524836.458 ਆਦਿ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।(Java) ਦੋ ਤਰ੍ਹਾਂ ਦੇ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦੀ ਹੈ: float ਅਤੇ double ਟਾਈਪਸ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਟਾਈਪਸ ਲਈ ਲੋੜੀਂਦੀਆਂ ਮੈਮੋਰੀ ਜ਼ਰੂਰਤਾਂ (Memory Requirement) ਅਤੇ ਉਹਨਾਂ ਵਿਚ ਸਟੋਰੇ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ਼ ਨੂੰ ਦਰਸਾ ਰਿਹਾ ਹੈ:

ਡਾਟਾ ਟਾਈਪ	ਸਟੋਰੇਜ਼ ਸਾਈਜ਼	ਡਾਟਾ ਦੀ ਰੇਂਜ਼
float	4 bytes	3. 4E-38 to 3. 4E+38
double	8 bytes	1.7E-308 to 1. 7E+308

ਟੇਬਲ 5.2 ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਡਾਟਾ ਦੀ ਸੰਖੇਪ ਵਿੱਚ ਜਾਣਕਾਰੀ

float ਡਾਟਾ ਟਾਈਪ ਇੱਕ ਸਿੰਗਲ-ਪ੍ਰੀਸੀਜ਼ਨ (single-precision) ਨੰਬਰ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ ਜਦੋਂ ਕਿ double ਟਾਈਪ ਡਬਲ-ਪ੍ਰੀਸੀਜ਼ਨ (double-precision) ਨੰਬਰ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਸਿੰਗਲ ਪ੍ਰੀਸੀਜ਼ਨ ਨੰਬਰ ਡਬਲ ਪ੍ਰੀਸੀਜ਼ਨ ਨਾਲੋਂ ਘੱਟ ਥਾਂ ਘੇਰਦਾ ਹੈ। ਜਦੋਂ ਸਟੋਰ ਕੀਤਾ ਜਾਣ ਵਾਲਾ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਮੁੱਲ ਜ਼ਿਆਦਾ ਵੱਡਾ ਹੋਵੇ ਤਾਂ ਉਸ ਨੂੰ ਸਿੰਗਲ- ਪ੍ਰੀਸੀਜ਼ਨ ਟਾਈਪ ਵਿਚ ਸਟੋਰ ਕਰਵਾਉਣ ਨਾਲ ਉਸ ਮੁੱਲ ਦੀ ਸ਼ੁੱਧਤਾ ਘੱਟ ਜਾਂਦੀ ਹੈ। ਇਸ ਲਈ ਜਦੋਂ ਸਾਨੂੰ ਵੱਡੇ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਮੁੱਲਾਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਦੀ ਲੋੜ ਪਵੇ ਤਾਂ ਡਬਲ ਟਾਈਪ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਸਭ ਤੋਂ ਵਧੀਆ ਆਪਸ਼ਨ ਰਹੇਗਾ। ਉਦਾਹਰਨ ਲਈ: ਜੇਕਰ ਅਸੀਂ ਵਿਦਿਆਰਥੀਆਂ ਦੁਆਰਾ ਪ੍ਰਾਪਤ ਅੰਕਾਂ ਦੇ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਤਾਂ ਫਲੋਟ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਇਸੇ ਤਰ੍ਹਾਂ, ਜਦੋਂ ਅਸੀਂ ਗਣਿਤਿਕ ਫੰਕਸ਼ਨਾਂ ਜਿਵੇਂ ਕਿ sin(), cos(), sqrt (),/ ਆਦਿ ਨਾਲ ਕੰਮ ਕਰ ਰਹੇ ਹੁੰਦੇ ਹਾਂ, ਤਾਂ ਸਾਨੂੰ ਡਬਲ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਕਰਨੀ ਚਾਹੀਦੀ ਹੈ। ਫਲੋਟ ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ 0.0f ਹੁੰਦਾ ਹੈ ਅਤੇ ਡਬਲ ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ 0.0d ਹੁੰਦਾ ਹੈ।

III. ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ (Character Data Type) : ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਸਿੰਗਲ ਕੋਟ (') ਵਿੱਚ ਬੰਦ ਇੱਕ ਸਿੰਗਲ ਯੂਨੀਕੋਡ ਅੱਖਰ (single Unicode character enclosed in single quotes) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਦਰਸਾਉਣ ਲਈ char ਕੀਵਰਡ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ 2 ਬਾਈਟ (16 ਬਿੱਟ) ਮੈਮੋਰੀ ਸਪੇਸ ਲੈਂਦਾ ਹੈ। char ਡਾਟਾ ਟਾਈਪ ਦੀ ਰੇਂਜ 0 ਤੋਂ 65535 ਹੁੰਦੀ ਹੈ। char ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ '/u0000' ਹੁੰਦਾ ਹੈ।

ਡਾਟਾ ਟਾਈਪ	ਸਟੋਰੇਜ਼ ਸਾਈਜ਼	ਡਾਟਾ ਦੀ ਰੇਂਜ਼
char	2 bytes	0 to 65535

ਟੇਬਲ 5.3 ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਸੰਖੇਪ ਵਿੱਚ ਵਿਆਖਿਆ

IV. ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ Boolean Data Type : ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲਾਂ ਵਿਚ ਬੁਲੀਅਨ ਮੁੱਲਾਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਡਾਟਾ ਟਾਈਪ ਸਿਰਫ਼ ਦੋ ਮੁੱਲਾਂ ਨੂੰ ਸਵੀਕਾਰ ਕਰਦਾ ਹੈ: true ਜਾਂ false ਮੁੱਲ। ਕੀਵਰਡ (boolean) ਦੀ ਵਰਤੋਂ ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ 1-ਬਿੱਟ ਜਾਣਕਾਰੀ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਪਰ ਇਸਦਾ ਮੈਮਰੀ ਆਕਾਰ ਸਹੀ ਰੂਪ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ। ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ false ਹੁੰਦਾ ਹੈ। ।

5.1.2 ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Non-Primitive Data Types) : ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਯੂਜ਼ਰ ਦੁਆਰਾ ਪਰਿਭਾਸ਼ਿਤ User defined ਡਾਟਾ ਟਾਈਪਸ ਜਾਂ ਰੈਫਰੈਂਸ Raference ਡਾਟਾ ਟਾਈਪਸ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਡਾਟਾ ਟਾਈਪਸ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਤੋਂ ਹੀ ਤਿਆਰ ਕੀਤੀਆਂ ਗਈਆਂ ਹੁੰਦੀਆਂ ਹਨ। ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ ਅਤੇ ਐਰੇ ਜਾਵਾ ਵਿੱਚ ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ।

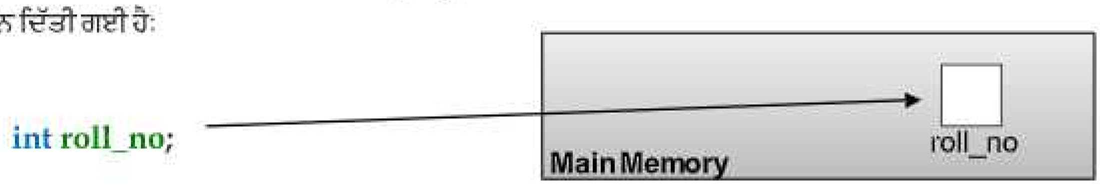
5.2 ਵੇਰੀਏਬਲਜ਼ (VARIABLES)

ਵੇਰੀਏਬਲ ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ (memory location) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਇਹ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ ਡਾਟਾ/ਮੁੱਲ ਸਟੋਰ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਲੋਕੇਸ਼ਨਾਂ ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਡਾਟਾ/ਮੁੱਲ ਨੂੰ ਵੇਰੀਏਬਲ ਦੇ ਨਾਮ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਐਕਸੈੱਸ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਦੀ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਸਮੇਂ (execution time) ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹਰੇਕ ਵੇਰੀਏਬਲ ਦੀ ਇੱਕ ਡਾਟਾ ਟਾਈਪ ਹੁੰਦੀ ਹੈ, ਜੋ ਵੇਰੀਏਬਲ ਦੇ ਮੈਮਰੀ ਆਕਾਰ ਅਤੇ ਉਸ ਵਿਚ ਸਟੋਰ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲ ਦੀ ਕਿਸਮ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੇਰੀਏਬਲਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਉਹਨਾਂ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਡਿਕਲੇਅਰ/ਘੋਸ਼ਿਤ (declare) ਕਰਨਾ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ।

5.2.1 ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ (Variable Declaration) : ਜਾਵਾ ਇੱਕ ਸਟ੍ਰਾਂਗਲੀ ਟਾਈਪਡ ਭਾਸ਼ਾ (Stgrongly Typed Language) ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੁੰਦਾ ਹੈ ਕਿ ਸਾਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੇਰੀਏਬਲਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਉਹਨਾਂ ਨੂੰ ਡਿਕਲੇਅਰ/ਘੋਸ਼ਿਤ ਕਰਨਾ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ। ਜੇਕਰ ਅਸੀਂ ਵੇਰੀਏਬਲਾਂ ਨੂੰ ਡਿਕਲੇਅਰ ਕੀਤੇ ਬਿਨਾਂ ਵਰਤਦੇ ਹਾਂ, ਤਾਂ ਕੰਪਾਈਲਰ ਸਿੰਟੈਕਸ ਐਰਰ (Syntax Error) ਦਿਖਾਵੇਗਾ। ਇੱਕ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਵਿੱਚ ਦੋ ਭਾਗ ਹੁੰਦੇ ਹਨ: ਡਾਟਾ ਟਾਈਪ ਅਤੇ ਵੇਰੀਏਬਲ ਦਾ ਨਾਮ (ਆਈਡੈਂਟੀਫਾਇਰ)। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੇਰੀਏਬਲ ਡਿਕਲੇਅਰ ਕਰਨ ਲਈ ਹੇਠਾਂ ਦਿੱਤੇ ਸਿੰਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ:

```
data_type variable_name;
```

ਇੱਥੇ data-name ਕੰਪਾਈਲਰ ਨੂੰ ਇਹ ਦੱਸਦਾ ਹੈ ਕਿ ਵੇਰੀਏਬਲ ਵਿੱਚ ਕਿਸ ਕਿਸਮ ਦਾ ਮੁੱਲ ਸਟੋਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਅਤੇ (variable-name) ਇੱਕ ਵੈਧ ਆਈਡੈਂਟੀਫਾਇਰ (valid indentifier) ਹੈ ਜੋ ਕੰਪਾਈਲਰ ਨੂੰ ਵੇਰੀਏਬਲ ਦੇ ਨਾਮ ਬਾਰੇ ਦੱਸਦਾ ਹੈ ਅਤੇ ਜਿਸਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਮੁੱਲ ਨੂੰ ਵਰਤਣ ਸਮੇ ਕੀਤੀ ਜਾਵੇਗੀ। ਹੇਠਾਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਦੀ ਇੱਕ ਉਦਾਹਰਨ ਦਿੱਤੀ ਗਈ ਹੈ:



ਚਿੱਤਰ : 5.2 ਵੇਰੀਏਬਲ ਦੀ ਮੈਮਰੀ ਐਲੋਕੇਸ਼ਨ

ਇਸ ਉਦਾਹਰਣ ਵਿਚ int ਇੰਟੀਜਰ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜੋ ਆਈਡੈਂਟੀਫਾਇਰ roll_no ਨੂੰ 4 ਬਾਈਟ ਮੈਮੋਰੀ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਇਹ ਆਈਡੈਂਟੀਫਾਇਰ ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਦੀ ਵਰਤੋਂ ਸਮੇਂ ਉਸ ਵਿੱਚ ਸਟੋਰ ਮੁੱਲ ਨੂੰ ਐਕਸੈਸ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਵੇਗਾ। ਇਸ ਉਦਾਹਰਣ ਅਨੁਸਾਰ ਵੇਰੀਏਬਲ roll_no ਸਿਰਫ਼ ਪੂਰਨ ਅੰਕ ਮੁੱਲ ਹੀ ਸਟੋਰ ਕਰ ਸਕਦਾ ਹੈ। ਅਸੀਂ ਕਾਮੇ ਸੇਪਰੇਟਰ (comma separator) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕੋ ਡਾਟਾ ਟਾਈਪ ਵਾਲੇ ਕਈ ਵੇਰੀਏਬਲ ਵੀ ਡਿਕਲੇਅਰ ਕਰ ਸਕਦੇ ਹਾਂ। ਉਦਾਹਰਣ ਲਈ:

```
int a, b, c;
```

5.2.2 ਵੇਰੀਏਬਲ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ੇਸ਼ਨ (Variable Initialization):

ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਵਿਚ ਉਸ ਨੂੰ ਸਿਰਫ ਮੈਮੋਰੀ ਐਲੋਕੇਟ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਉਸ ਵਿਚ ਕੋਈ ਡਾਟਾ ਸਟੋਰ ਨਹੀਂ ਕੀਤਾ ਜਾਂਦਾ। ਅਸੀਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਸਮੇਂ ਉਸ ਨੂੰ ਇੱਕ ਮੁੱਲ ਵੀ ਅਸਾਈਨ (assign) ਕਰ ਸਕਦੇ ਹਾਂ, ਜਿਸਨੂੰ ਵੇਰੀਏਬਲ ਇਨੀਸ਼ੀਅਲਾਈਜੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ:

```
int roll_no=5;
```

ਚਿੱਤਰ : 5.3 ਵੇਰੀਏਬਲ ਵਿੱਚ ਮੁੱਲ ਸਟੋਰ ਕਰਨਾ

ਇਸ ਅਸਾਈਨ ਕੀਤੇ ਮੁੱਲ ਨੂੰ ਬਾਅਦ ਵਿੱਚ ਕਿਸੇ ਵੀ ਸਮੇਂ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ ਕਿਉਂਕਿ ਇੱਕ ਵੇਰੀਏਬਲ ਸਾਨੂੰ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੌਰਾਨ (during execution) ਕਿਸੇ ਵੀ ਸਮੇਂ ਇਸਦੇ ਮੁੱਲ ਨੂੰ ਬਦਲਣ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ।

5.2.3 ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਦੀ ਵਰਤੋਂ ਕਰਨਾ (Accessing Value of a Variable) :

ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਮੁੱਲ ਨੂੰ ਐਕਸੈਸ ਕਰਨ/ਵਰਤਣ ਲਈ ਸਾਨੂੰ ਵੇਰੀਏਬਲ ਦਾ ਨਾਮ ਵਰਤਣਾ ਪੈਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ

```
int num1=5;
int num2 = num1 + num1;
System.out.println(num2);
```

ਉਪਰੋਕਤ ਉਦਾਹਰਣ ਵਿੱਚ ਅਸੀਂ ਦੋ ਵੇਰੀਏਬਲ num1 ਅਤੇ num2 ਨੂੰ ਡਿਕਲੇਅਰ ਅਤੇ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ ਕੀਤਾ ਹੈ। num1 ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਲ num 2 ਦੇ ਮੁੱਲ ਦੀ ਗਣਨਾ (int num2 = num1 + num1) ਕਰਨ ਲਈ ਵਰਤਿਆ ਗਿਆ ਹੈ।।num2 ਦਾ ਮੁੱਲ ਦਿਖਾਉਣ ਲਈ ਅਸੀਂ ਜਾਵਾ ਦੇ println () ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹਾਂ।

5.3 ਆਪਰੇਟਰਜ਼ (OPERATORS)

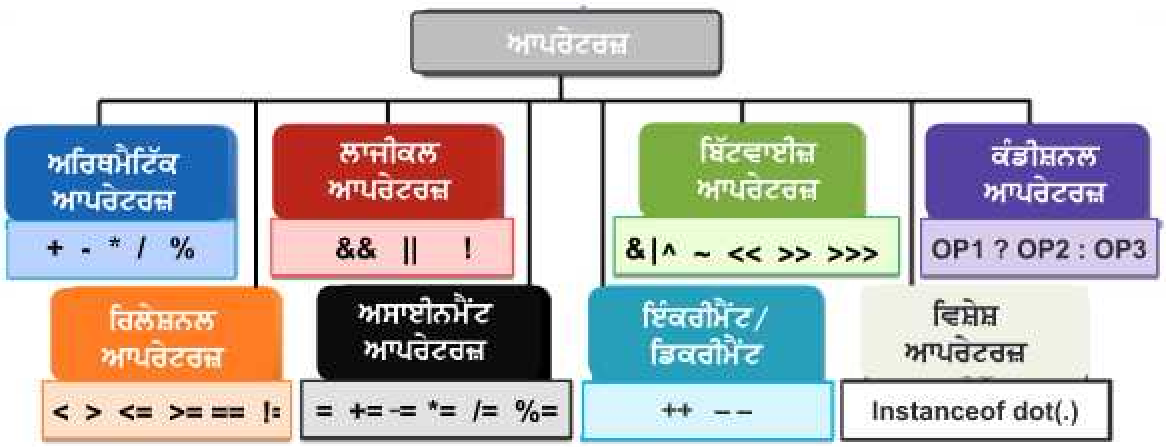
ਆਪਰੇਟਰਜ਼ ਉਹ ਚਿੰਨ੍ਹ ਹੁੰਦੇ ਹਨ ਜਿਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਡਾਟਾ ਉਪਰ ਕਿਸੇ ਖਾਸ ਆਪਰੇਸ਼ਨ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਲਈ: + ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਜੋੜ ਕਰਨ ਲਈ, * ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਗੁਣਾ ਕਰਨ ਲਈ, >= ਦੀ ਵਰਤੋਂ ਤੁਲਨਾ ਕਰਨ ਲਈ। ਇਹਨਾਂ ਉਦਾਹਰਣਾਂ ਵਿਚ +, *, >= ਆਪਰੇਟਰਜ਼ ਹਨ ਜੋ ਕਿ ਵੱਖ-ਵੱਖ ਆਪਰੇਸ਼ਨਾਂ (ਕੰਮਾਂ) ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰਨ ਤੋਂ ਬਾਅਦ ਇਕ ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੇ ਹਨ।

ਕਿਸੇ ਵੀ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਸਾਨੂੰ ਆਪਰੈਂਡਜ਼ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਆਪਰੈਂਡਜ਼ ਉਹ ਡਾਟਾ ਆਈਟਮਾਂ ਹੁੰਦੀਆਂ ਹਨ

ਜਿਨ੍ਹਾਂ ਉਪਰ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰ ਸਕਦੇ ਹੁੰਦੇ ਹਨ। ਇਹ ਆਪਰੇਂਡ ਵੇਰੀਏਬਲਜ਼ ਜਾਂ ਕਾਂਸਟੈਂਟ ਮੁੱਲ ਕੁੱਝ ਵੀ ਹੋ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਣ ਲਈ:

a + 5 * 10

ਇਸ ਉਦਾਹਰਣ ਵਿਚ + ਅਤੇ * ਆਪਰੇਟਰਜ਼ ਹਨ ਜੋ ਆਪਣਾ ਕੰਮ ਵੇਰੀਏਬਲ 'a' ਕਾਂਸਟੈਂਟ ਮੁੱਲ 5 ਅਤੇ 10 ਉਪਰ ਕਰ ਰਹੇ ਹਨ। ਇਸ ਵਿਚ 'a', 5 ਅਤੇ 10 ਨੂੰ ਆਪਰੇਂਡ ਕਿਹਾ ਜਾਵੇਗਾ। ਓਪਰੇਟਰਾਂ ਅਤੇ ਓਪਰੇਡਾਂ ਦੇ ਇੱਕ ਵੈਲੀਡ ਸਮੂਹ (valid combination) ਨੂੰ ਐਸਪ੍ਰੈਸ਼ਨ/ਸਮੀਕਰਨ (Expression) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਆਪਰੇਟਰਾਂ ਦੁਆਰਾ ਕੀਤੇ ਜਾਂਦੇ ਫੰਕਸ਼ਨਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਜਾਵਾ ਆਪਰੇਟਰਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ: ਅਰਿਥਮੈਟਿੱਕ (Arithmetic) ਆਪਰੇਟਰਜ਼, ਰਿਲੇਸ਼ਨਲ (Relational) ਆਪਰੇਟਰਜ਼, ਲਾਜ਼ੀਕਲ (Logical) ਆਪਰੇਟਰਜ਼, ਅਸਾਈਨਮੈਂਟ (Assignment) ਆਪਰੇਟਰਜ਼, ਬਿਟਵਾਈਜ਼ (Bitwise) ਆਪਰੇਟਰਜ਼, ਇਨਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Increment and Decrement) ਆਪਰੇਟਰਜ਼, ਕੰਡੀਸ਼ਨਲ (Conditional) ਆਪਰੇਟਰਜ਼ ਅਤੇ ਸਪੈਸ਼ਲ (Special) ਆਪਰੇਟਰਜ਼।



ਚਿੱਤਰ 5.4 ਜਾਵਾ ਵਿੱਚ ਆਪਰੇਟਰਜ਼ ਦੀਆਂ ਕਿਸਮਾਂ

5.3.1 ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਜ਼ (Arithmetic Operators) :

ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਅਰਥਮੈਟਿਕ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕਿਤੀ ਜਾਂਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ: ਜੋੜ, ਘਟਾਓ, ਗੁਣਾ ਭਾਗ ਆਦਿ। ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ 5 ਅਰਥਮੈਟਿਕ ਆਪਰੇਟਰ ਹਨ। ਇਹ ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਬਾਈਨਰੀ ਕਿਸਮ ਦੇ ਆਪਰੇਟਰ ਹਨ ਕਿਉਂਕਿ ਇਹਨਾਂ ਸਾਰੇ ਆਪਰੇਟਰਾਂ ਨੂੰ ਆਪਣਾ ਕੰਮ ਕਰਨ ਲਈ ਦੋ ਆਪਰੈਂਡਜ਼ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਹੇਠਾਂ ਦਿਤਾ ਟੇਬਲ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੇ ਕੰਮ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਨਾਮ	ਆਪਰੇਟਰ	ਵਿਆਖਿਆ	ਆਪਰੇਟਰ ਦੀ ਉਦਾਹਰਣ	
			ਇੰਟੀਜ਼ਰ ਮੁੱਲ	ਰੀਅਲ ਮੁੱਲ
ਐਡ (Add)	+	ਜੋੜ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	2+4→6	2.0+4.0 →6.0
ਸਬਟ੍ਰੈਕਟ (Subtract)	-	ਘਟਾਓ ਕਰਨ ਲਈ ਜਾਂ ਯੂਨਰੀ ਮਾਈਨਟਸ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹਹੈ।	6-2→4	6.0-4.0 → 2.0
ਮਲਟੀਪਲਾਈ (Multiply)	*	ਗੁਣਾ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	7*2→14	7.0*2.0 → 14.0
ਡਿਵਾਈਡ (Divide)	/	ਭਾਗ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	5/2→2	5.0/2.0 →2.5
ਮਾਡੂਲਸ (Modulus)	%	ਭਾਗ ਕਰਨ ਤੋਂ ਬਾਅਦ ਬਾਕੀ ਬਚੇ ਮੁੱਲ (ਸ਼ੇਸ਼ਫਲ) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	7%4→3	5.0%2.0 → 1.0

ਟੇਬਲ 5.4 ਅਰਿਥਮੈਟਿਕ ਆਪ੍ਰੇਟਰਜ਼

ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿਚ ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਪ੍ਰੋਗਰਾਮ 5.1: ਜਾਵਾ ਵਿਚ ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ

Test1.java

```java
class Test1
{
    public static void main(String args[])
    {
        float num1=5.0f, num2=2.0f, result;
        result=num1+num2;
        System.out.println("Result of Addition is "+result);
        result=num1-num2;
        System.out.println("Result of Subtraction is "+result);
        result=num1*num2;
        System.out.println("Result of Multiplication is "+result);
        result=num1/num2;
        System.out.println("Result of Division is "+result);
        result=num1%num2;
        System.out.println("Result of Modulus is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5.1 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Exeuction) ਅਤੇ ਆਊਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac Test1.java

D:\Java Programs>java Test1
Result of Addition is 7.0
Result of Subtraction is 3.0
Result of Multiplication is 10.0
Result of Division is 2.5
Result of Modulus is 1.0

D:\Java Programs>
```

ਪ੍ਰੋਗਰਾਮ 5.1 ਵਿੱਚ ਅਸੀਂ ਫਲੋਟਿੰਗ ਟਾਈਪ ਦੇ ਤਿੰਨ ਵੇਰੀਏਬਲ ਡਿਕਲੇਅਰ ਕੀਤੇ ਹਨ: num1, num2 ਅਤੇ result ਵੇਰੀਏਬਲਜ਼। ਵੇਰੀਏਬਲ num1 ਅਤੇ num2 ਨੂੰ ਕ੍ਰਮਵਾਰ ਫਲੋਟ ਮੁੱਲ 5.0f ਅਤੇ 2.0f ਨਾਲ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ ਕੀਤਾ ਹੈ। ਫਿਰ ਸਾਰੇ ਅਰਥਮੈਟਿੱਕ ਆਪਰੇਟਰਾਂ ਦੀ ਗਣਨਾ ਦਾ ਨਤੀਜਾ result ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਹੈ ਜਿਸ ਨੂੰ ਜਾਵਾ ਦੇ System.out.println( ) ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਪ੍ਰਦਰਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ। ਅਸੀਂ ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਨੂੰ ਇੰਟੀਜ਼ਰ ਅਤੇ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਆਪਰੇਂਡਾਂ 'ਤੇ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹਾਂ। ।

5.3.2 ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ (Relational Operators)

ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਤੁਲਨਾਤਮਕ (comparison) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਆਪਰੈਂਡਜ਼ ਵਿਚਕਾਰ ਰਿਲੇਸ਼ਨਸ਼ਿਪ ਟੈਸਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਦੂਸਰੇ ਸ਼ਬਦਾਂ ਵਿਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਮੁੱਲਾਂ ਦੀ ਤੁਲਨਾ (compare) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਮੁੱਲਾਂ ਦੀ ਤੁਲਨਾ ਕਰਨ ਤੋਂ ਬਾਅਦ ਇਹ ਆਪਰੇਟਰ ਸਾਨੂੰ true ਜਾਂ false ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੇ ਹਨ। ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ ਇਹ ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਬਾਈਨਰੀ ਕਿਸਮ ਦੇ ਆਪਰੇਟਰਜ਼ ਹਨ। ਭਾਵ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ ਦੋ ਆਪਰੇਂਡਜ਼ ਦੀ ਜ਼ਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਜਾਵਾ ਵਿਚ ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ ਦੀ ਗਿਣਤੀ 6 ਹੈ ਜੋ ਕਿ ਹੇਠਾਂ ਟੇਬਲ ਵਿਚ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਦਿੱਤੇ ਗਏ ਹਨ:

ਨਾਂ	ਆਪਰੇਟਰ	ਵਿਆਖਿਆ	ਉਦਾਹਰਣ	ਨਤੀਜਾ
ਬਰਾਬਰ ਨਹੀਂ (Equals to)	==	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਦੋ ਮੁੱਲ ਬਰਾਬਰ ਹਨ	4==5 5==5	False True
ਬਰਾਬਰ ਨਹੀਂ ਹੈ (Not Equals to)	!=	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਦੋ ਮੁੱਲ ਬਰਾਬਰ ਨਹੀਂ ਹੈ	4! =5 4! =4	True False
ਵੱਡਾ ਹੈ (Greater than)	>	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲਾ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਛੋਟਾ ਹੈ	4>5 5>4	False True
ਛੋਟਾ ਹੈ (Less than)	<	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲਾ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਵੱਡਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ।	4<5 5<4	True False
ਵੱਡਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ (Greater than or equal to)	>=	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲੇ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਵੱਡਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ।	5>=5 6>=8 10>=5	True False True
ਛੋਟਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ (Less than or equal to)	<=	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲਾ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਛੋਟਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ।	4<=5 4<=2 4<=4	True False True

ਟੇਬਲ 5.5 ਜਾਵਾ ਵਿੱਚ ਰਿਲੇਸ਼ਨਸ

ਪ੍ਰੋਗਰਾਮ 5 . 2 : ਜਾਵਾ ਵਿੱਚ ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ

```
class Test2
{
    public static void main(String args[])
    {
        int num1=5, num2=2;
        boolean result;
        System.out.println("Value of num1 is "+num1);
        System.out.println("Value of num2 is "+num2);
        result=num1==num2;
        System.out.println("Result of num1==num2 is "+result);
        result=num1!=num2;
        System.out.println("Result of num1!=num2 is "+result);
        result=num1>num2;
        System.out.println("Result of num1>num2 is "+result);
        result=num1<num2;
        System.out.println("Result of num1<num2 is "+result);
        result=num1>=num2;
        System.out.println("Result of num1>=num2 is "+result);
        result=num1<=num2;
        System.out.println("Result of num1<=num2 is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5.2 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ: ◆on) ਅਤੇ ਆਊਟਪੁੱਟ:

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac Test2.java

D:\Java Programs>java Test2
Value of num1 is 5
Value of num2 is 2
Result of num1==num2 is false
Result of num1!=num2 is true
Result of num1>num2 is true
Result of num1<num2 is false
Result of num1>=num2 is true
Result of num1<=num2 is false

D:\Java Programs>_
```

ਆਮ ਤੌਰ 'ਤੇ ਰਿਲੇਸ਼ਨਲ ਓਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਉਹਨਾਂ ਟੈਸਟ ਐਕਸਪੈਸ਼ਨਜ਼ ਨੂੰ ਬਨਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜਿਹਨਾਂ ਦੀ ਵਰਤੋਂ ਬ੍ਰਾਂਚਿੰਗ, ਲੂਪਿੰਗ ਅਤੇ ਜੰਪਿੰਗ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਵਿੱਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਨ੍ਹਾਂ ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਿਆਖਿਆ ਇਸ ਪੁਸਤਕ ਦੇ ਅਗਲੇ ਪਾਠ ਵਿਚ ਕੀਤੀ ਗਈ ਹੈ ॥

5.3.3 ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ (Logical Operators): ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਬੂਲੀਅਨ (Boolean) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਜ ਦੀ ਵਰਤੋਂ ਮਿਸ਼ਰਿਤ (compound) ਰਿਲੇਸ਼ਨਲ ਐਕਸਪੈਸ਼ਨਜ਼ ਬਨਾਉਣ ਲਈ ਕਿਤੀ ਜਾਂਦੀ ਹੈ। ਦੂਸਰੇ ਸ਼ਬਦਾਂ ਵਿਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਉਸ ਸਮੇਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਅਸੀਂ ਇਕ ਸਮੇਂ ਵਿਚ ਇਕ ਤੋਂ ਵੱਧ ਟੈਸਟ ਕੰਡੀਸ਼ਨਾਂ ਦੀ ਜਾਂਚ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋਈਏ। ਸੀ ਭਾਸ਼ਾ ਵਿਚ 3 ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਮੌਜੂਦ ਹਨ: 'ਲਾਜੀਕਲ AND, "ਲਾਜੀਕਲ OR" ਅਤੇ 'ਲਾਜੀਕਲ NOT', ਇਹਨਾਂ ਵਿਚ "ਲਾਜੀਕਲ AND" ਅਤੇ 'ਲਾਜੀਕਲ OR' ਬਾਈਨਰੀ ਆਪਰੇਟਰਜ਼ ਹਨ ਜਦੋਂ ਕਿ 'ਲਾਜੀਕਲ NOT' ਯੂਨਰੀ ਆਪਰੇਟਰ ਹੈ। 'ਲਾਜੀਕਲ AND' ਅਤੇ ਲਾਜੀਕਲ OR ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ ਦੋ ਆਪਰੈਂਡਜ਼ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ ਜਦੋਂ ਕਿ NOT ਆਪ੍ਰੇਟਰ ਨੂੰ ਇਕ ਆਪਰੇਂਡ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਸਾਰੇ ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ true ਜਾਂ false ਨਤੀਜਾ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਲਾਜੀਕਲ AND ਆਪ੍ਰੇਟਰ true ਨਤੀਜਾ ਕੇਵਲ ਉਸ ਸਮੇਂ ਦੇਵੇਗਾ ਜਦੋਂ ਇਸਦੇ ਦੋਵੇਂ ਆਪਰੈਂਡਜ਼ true ਹੋਣ, ਇਸੇ ਤਰ੍ਹਾਂ ਲਾਜੀਕਲ OR ਆਪਰੇਟਰ ਦਾ true ਨਤੀਜਾ ਕੇਵਲ ਉਸ ਸਮੇਂ ਆਵੇਗਾ ਜਦੋਂ ਇਸਦਾ ਜਾਂ ਤਾਂ ਕੋਈ ਵੀ ਇਕ ਆਰੇਂਡ true ਹੋਵੇ ਜਾਂ ਦੋਵੇਂ ਆਪਰੈਂਡਜ਼ true ਹੋਣ। ਲਾਜੀਕਲ NOT ਆਪਰੇਟਰ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾ ਜੇਕਰ ਇਸਦਾ ਐਕਸਪ੍ਰੈਸ਼ਨ false ਹੋਵੇ। ਹੇਠਾਂ ਦਿੱਤੇ ਟੇਬਲ ਵਿਚ ਜਾਵਾ ਵਿਚ ਮੌਜੂਦ ਸਾਰੇ ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਢੁਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਦਰਸ਼ਾਇਆ ਗਿਆ ਹੈ:

<table>
<tr><th>ਨਾਂ</th><th>ਆਪਰੇਟਰ</th><th>ਵਿਆਖਿਆ</th><th>ਉਦਾਹਰਣ</th><th>ਵਿਆਖਿਆ ਅਤੇ ਨਤੀਜਾ</th></tr>
<tr><td>AND</td><td>&&</td><td>ਇਹ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾਜ ਦੇ ਦੋਵੇਂ ਅਕਸਪ੍ਰੈਸ਼ਨ true ਹੋਣ ਨਤੀਜਾ ਨਤੀਜਾ false ਆਵੇਗਾ।</td><td>3>5 && 4>5
3>5 && 4<5
3<5 && 4>5
3<5 && 4<5</td><td>False && False → False
False && True → False
True && False → False
True && True → True</td></tr>
<tr><td>OR</td><td>||</td><td>ਇਹ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾ ਜਦੋ ਦੋਵੇਂ ਅਕਸਪ੍ਰੈਸ਼ਨਾਂ ਵਿੱਚੋ ਘੱਟ ਤੋਂ ਘੱਟ ਇਕ ਅਕਸਪ੍ਰੈਸ਼ਨ true ਹੋਵੇ ਨਹੀਂ ਤਾਂ ਨਤੀਜਾ false ਆਵੇਗਾ।</td><td>3>5 || 4>5
3>5 || 4<5
3<5 || 4>5
3<5 || 4<5</td><td>False || False → False
False || True → True
True || False → True
True || True → True</td></tr>
<tr><td>NOT</td><td>!</td><td>ਇਹ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾ ਜਦੋ ਇਸਦਾ ਅਕਸਪ੍ਰੈਸ਼ਨ false ਹੋਵੇਗਾ ਨਹੀਂ ਤਾਂ ਨਤੀਜਾ false ਆਵੇਗਾ।</td><td>!(3<5)
!(3>5)</td><td>!(True) → False
!(False) → True</td></tr>
</table>

ਟੇਬਲ 5.4 ਜਾਵਾ ਵਿੱਚ ਲਾਜੀਕਲ ਆਪ੍ਰੇਟਰਜ਼

ਪ੍ਰੋਗਰਾਮ 5.3 ਜਾਵਾ ਵਿਚ ਲਾਜੀਕਲ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ

```
class Test3
{
    public static void main(String args[])
    {
        int num1=15, num2=10, num3=5;
        boolean result;
        System.out.println("Value of num1 is "+num1);
        System.out.println("Value of num2 is "+num2);
        System.out.println("Value of num3 is "+num3);
        result=num1>num2 && num2<=num3;
        System.out.println("Result of AND (num1>num2 && num2<=num3) is "+result);
        result=num1>num2 || num2<=num3;
        System.out.println("Result of OR (num1>num2 || num2<=num3) is "+result);
        result=!(num1==num2);
        System.out.println("Result of NOT (!(num1==num2)) is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5 .3 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac Test3.java

D:\Java Programs>java Test3
Value of num1 is 15
Value of num2 is 10
Value of num3 is 5
Result of AND (num1>num2 && num2<=num3) is false
Result of OR (num1>num2 || num2<=num3) is true
Result of NOT (!(num1==num2)) is true

D:\Java Programs>
```

5.3.4 ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ (Assignment operators):

ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲ ਵਿਚ ਮੁੱਲ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦਾ ਚਿੰਨ੍ਹ = ਹੁੰਦਾ ਹੈ। ਹੇਠਾਂ ਦਿਤੀਆਂ ਉਦਾਹਰਣਾਂ ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦੀਆਂ ਹਨ :

```
int a = - 2;        // assigns –ve value (-2) to the variable.
int b = 5;          // assigns value (5) to the variable.
int c = a + b;      // assigns the result of expression to the variable.
int a = a + 10;     // self-assignment of a variable.
```

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੇ ਖੱਬੇ ਪਾਸੇ ਹਮੇਸ਼ਾਂ ਇੱਕ ਵੈਧ ਆਈਡੈਂਟੀਫਾਇਰ (valid Indentifier) ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੋਵੇ। ਅਸੀਂ ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੇ ਖੱਬੇ ਪਾਸੇ ਐਕਸਪ੍ਰੇਸ਼ਨ/ਸਮੀਕਰਨ ਨਹੀਂ ਲਿੱਖ ਸਕਦੇ। ਉਦਾਹਰਨ ਲਈ ਨਿਮਨਲਿਖਤ ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟ ਗਲਤ ਹੋਵੇਗੀ:

```
a +b=c;             //Invalid assignment statement
```

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੀ ਮਦਦ ਨਾਲ ਕਈ ਵੇਰੀਏਬਲਾਂ ਨੂੰ ਇੱਕ ਸਾਂਝਾ ਮੁੱਲ ਵੀ ਨਿਰਧਾਰਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ:

```
a = b = c = 5;      //value 5 will be assigned to three variables a, b and c
```

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕੰਪਾਊਂਡ ਜਾਂ ਸ਼ਾਰਟਹੈਂਡ ਅਸਾਈਨਮੈਂਟ (Compound or Shorthand Assignment) ਆਪਰੇਟਰਜ਼ ਵਜੋਂ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਸ਼ਾਰਟਹੈਂਡ ਆਪਰੇਟਰਜ਼ ਸੈਲਫ-ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟਜ਼ ਵਿਚ ਲਾਭਦਾਇਕ ਹੁੰਦੇ ਹਨ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਸ਼ਾਰਟਹੈਂਡ ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਮੰਨ ਲਵੋ int a=5;

ਸ਼ਾਰਟਹੈਂਡ ਆਪਰੇਟਰ	ਉਦਾਹਰਣ	ਵਿਆਖਿਆ	ਨਤੀਜਾ
+=	a+ =2	a = a + 2	a=7
- =	a- =2	a = a – 2	a=3
*=	a* =2	a = a * 2	a=10
/=	a / =2	a = a / 2	a=2
%=	a%=2	a = a % 2	a=1

ਟੇਬਲ 5.7 ਅਰਥਮੈਟਿਕ ਆਪ੍ਰੇਟਰ ਲਈ ਸ਼ਾਰਟਹੈਂਡ ਅਸਾਇਨਮੈਂਟ

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ "=" ਅਤੇ Equals to ਆਪਰੇਟਰ "==" ਦੋਵੇਂ ਵੱਖ-ਵੱਖ ਆਪਰੇਟਰਜ਼ ਹਨ ।। ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ (=) ਦੀ ਵਰਤੋਂ ਕਿਸੇ variable ਵਿਚ ਮੁੱਲ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਕਿ ਇਕੁਐਲਟੀ (= =) (equality ਭਾਵ equals to) ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਇਹ ਤੈਅ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਦਿੱਤੇ ਗਏ ਆਪਰੈਂਡਜ਼ ਦਾ ਮੁੱਲ ਬਰਾਬਰ ਹੈ ਜਾਂ ਨਹੀਂ। ਇਹ ਦੋਵੇਂ ਆਪਰੇਟਰਜ਼ ਇਕ ਦੂਜੇ ਦੀ ਜਗ੍ਹਾਂ ਨਹੀਂ ਵਰਤੇ ਜਾ ਸਕਦੇ।

5.3.5 ਬਿੱਟਵਾਈਜ਼ ਆਪਰੇਟਰਜ਼ (Bitwise Operators):

ਬਿੱਟਵਾਈਜ਼ ਓਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਬਿੱਟ ਲੇਵਲ ਆਪਰੇਸ਼ਨ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹਨਾਂ ਨੂੰ ਕਿਸੇ ਵੀ ਇੰਟੀਜ਼ਰ ਟਾਈਪ (char, short, int, ਆਦਿ) ਮੁੱਲਾਂ ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਬਿੱਟਵਾਈਜ਼ ਆਪਰੇਟਰਾਂ ਸੰਬੰਧੀ ਸੰਖੇਪ ਜਾਣਕਾਰੀ ਦਰਸਾ ਰਿਹਾ ਹੈ:

ਮੰਨ ਲਵੋ **int A=60** ਅਤੇ **int B=13** ਤਾਂ :

Operator	Description	Example
& (Bitwise AND)	ਇਹ ਆਪਰੇਅਰ ਇੱਕ ਬਾਈਨਰੀ ਓਪਰੇਟਰ ਹੈ,ਇਹ ਇਨਪੁੱਟ ਮੁੱਲਾਂ ਉੱਪਰ ਬਿੱਟ ਬਾਈ ਬਿੱਟ A©4 ਆਪਰੇਸ਼ਨ ਲਾਗੂ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਦੋਵੇਂ ਬਿੱਟ 1 ਹਨ ਤਾਂ ਇਹ 1 ਦਿੰਦਾ ਹੈ, ਨਹੀ ਤਾਂ ਇਹ 0 ਵਾਪਿਸ ਕਰਨਾ।	(A & B) ਦਾ ਨਤੀਜਾ 12 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 000 1100 ਹੋਵੇਗਾ।
I (Bitwise OR)	ਇਹ ਆਪਰੇਟਰ ਇੱਕ ਬਾਈਨਰੀ ਓਪਰੇਟਰ ਹੈ,ਇਹ ਇਨਪੁੱਟ ਮੁੱਲਾਂ ਉੱਪਰ ਬਿੱਟ ਬਾਈ ਬਿੱਟ OR ਆਪਰੇਸ਼ਨ ਲਾਗੂ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਦੋਵੇਂ ਬਿੱਟ 1 ਹਨ ਤਾਂ ਇਹ 1 ਦਿੰਦਾ ਹੈ, ਨਹੀ ਤਾਂ ਇਹ 0 ਵਾਪਿਸ ਕਰਨਾ।	(A) (B) ਦਾ ਨਤੀਜਾ 61 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0011 1101 ਹੋਵੇਗਾ।
^ (Bitwise XOR)	ਇਹ ਆਪਰੇਟਰ ਇੱਕ ਬਾਈਨਰੀ ਓਪਰੇਟਰ ਹੈ,ਇਹ ਇਨਪੁੱਟ ਮੁੱਲਾਂ ਉੱਪਰ ਬਿੱਟ ਬਾਈ ਬਿੱਟ XOR ਆਪਰੇਸ਼ਨ ਲਾਗੂ ਕਰਦਾ ਹੈ ਜੇਕਰ ਸੰਬੰਧਿਤ ਬਿੱਟ ਵੱਖਰੇ ਹਨ ਤਾਂ ਇਹ 1 ਦਿੰਦਾ ਹੈ ਨਹੀਂ ਤਾਂ ਇਹ 0 ਵਾਪਿਸ ਕਰੇਗਾ।	(A) (B) ਦਾ ਨਤੀਜਾ 61 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0011 1101 ਹੋਵੇਗਾ।
~ (Bitwise Compliment)	ਇਹ 1's ਕੰਪਲੀਮੈਂਟ ਯੂਨਰੀ ਆਪਰੇਟਰ ਹੈ ਜੋ ਬਿੱਟਸ ਨੂੰ ਫਲਿਪ ਕਰਦਾ ਹੈ ਭਾਵ 0 ਨੂੰ 1 ਅਤੇ 1 ਨੂੰ ਵਿੱਚ ਤਬਦੀਲ ਕਰਦਾ ਹੈ।	(A^ B) ਦਾ ਨਤੀਜਾ 49 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0011 0001 ਹੋਵੇਗਾ।
<< (Right Shift)	ਇਹ ਲੈਫਟ ਸ਼ਿਫਟ ਆਪਰੇਟਰ ਹੈ। ਇਸ ਵਿੱਚ ਖੱਬੇ ਪਾਸੇ ਮੌਜੂਦ ਅਪਰੈਂਡ ਦੀਆਂ ਬਿੱਟਸ ਨੂੰ ਸੱਜੇ ਪਾਸੇ ਆਪਰੈਂਡ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਬਿੱਟਾਂ ਦੀ ਸੰਖਿਆ ਅਨੁਸਾਰ ਖੱਬੇ ਪਾਸੇ ਸ਼ਿਫਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।	(A) ਦਾ ਨਤੀਜਾ 61 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 2's ਕੰਪਲੀਮੈਂਟ ਰੂਪ ਵਿੱਚ 1100 0011 ਹੋਵੇਗਾ (ਕਿਉਂਕਿ ਨਤੀਜਾ signed binary number ਹੈ।)
>> (Right Shift)	ਇਹ ਰਾਈਟ ਸ਼ਿਫਟ ਆਪਰੇਟਰ ਹੈ। ਇਸ ਵਿੱਚ ਖੱਬੇ ਪਾਸੇ ਮੌਜੂਦ ਆਪਰੈਂਡ ਦੀਆਂ ਬਿੱਟਸ ਨੂੰ ਸੱਜੇ ਪਾਸੇ ਆਪਰੈਂਡ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਬਿੱਟਾਂ ਦੀ ਸੰਖਿਆ ਅਨੁਸਾਰ ਸੱਜੇ ਪਾਸੇ ਸ਼ਿਫਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।	A >> 2 ਦਾ ਨਤੀਜਾ 15 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0000 1111 ਹੋਵੇਗਾ।
>>> (Zero Fill Right Shift)	ਇਹ ਰਾਈਟ ਸ਼ਿਫਟ ਜ਼ੀਰੋ ਫਿਲ ਓਪਰੇਟਰ ਹੈ। ਇਸ ਵਿੱਚ ਖੱਬੇ ਪਾਸੇ ਮੌਜੂਦ ਆਪਰੈਂਡ ਬਿੱਟਸ ਨੂੰ ਸੱਜੇ ਆਪਰੈਂਡ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਬਿਟਾਂ ਦੀ ਸੰਖਿਆ ਅਨੁਸਾਰ ਸੱਜੇ ਪਾਸੇ ਸ਼ਿਫਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਸ਼ਿਫਟ ਕੀਤੇ ਮੁੱਲ ਜ਼ੀਰੋ ਨਾਲ ਭਰੇ ਜਾਂਦੇ ਹਨ।	A >>> 2 ਦਾ ਨਤੀਜਾ 15 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 00001111 ਹੋਵੇਗਾ।

ਟੇਬਲ 5.8 ਬਿਟਵਾਇਜ਼ ਆਪ੍ਰੇਟਰਜ਼

5.3.6 ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰਜ਼ (Increment and Decrement Operators):

ਇਹ ਯੂਨਰੀ ਆਪਰੇਟਰਜ਼ ਹਨ। ਚਿੰਨ ++ ਦੀ ਵਰਤੋਂ ਇੰਕਰੀਮੈਂਟ ਆਪਰੇਟਰ ਲਈ ਅਤੇ ਚਿੰਨ -- ਦੀ ਵਰਤੋਂ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰ ਵੱਜੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇੰਕਰੀਮੈਂਟ ਆਪਰੇਟਰ (++) ਆਪਣੇ ਆਪਰੈਂਡ ਦੇ ਮੁੱਲ ਵਿਚ ਇਕ ਨੰਬਰ ਦਾ ਵਾਧਾ ਕਰਦਾ ਹੈ ਜਦੋਂ ਕਿ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰ (- -) ਆਪਣੇ ਆਪਰੈਂਡ ਦੇ ਮੁੱਲ ਵਿਚੋਂ ਇਕ ਨੰਬਰ ਘਟਾਅ ਦਿੰਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਨਾਲ ਵਰਤਿਆ ਜਾਣ ਵਾਲਾ ਆਪਰੈਂਡ ਇਕ ਵੇਰੀਏਬਲ ਹੀ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ। ਇਹਨਾਂ ਨੂੰ ਕਿਸੇ ਸਥਿਰ ਮੁੱਲ ਉਪਰ ਸਿੱਧੇ ਲਾਗੂ ਨਹੀਂ ਕਿਹਾ ਜਾ ਸਕਦਾ॥

ਉਦਾਹਰਣ ਲਈ:

int x=10;   (x ਇਕ ਇੰਟੀਜ਼ਰ ਵੇਰੀਏਬਲ ਹੈ ਜਿਸਦਾ ਮੁੱਲ ਹੈ 10)

ਹੇਠਾਂ ਦਿਤੀ ਐਕਸਪ੍ਰੈਸ਼ਨ x ਦੇ ਮੁੱਲ ਨੂੰ ਵਧਾ ਕੇ 11 ਕਰ ਦੇਵੇਗੀ।

++x;   (ਜੋ ਕਿ x= x+1 ਦੇ ਬਰਾਬਰ ਹੈ)

ਇਸੇ ਤਰ੍ਹਾਂ, ਹੇਠਾਂ ਦਿੱਤੀ ਐਕਸਪ੍ਰੈਸ਼ਨ x ਦੇ ਅਸਲੀ ਮੁੱਲ (10) ਨੂੰ ਘਟਾਅ ਕੇ 9 ਕਰ ਦੇਵੇਗੀ।

--x,   (ਜੋ ਕਿ x=x-1 ਦੇ ਬਰਾਬਰ ਹੈ)

ਜੇਕਰ ਅਸੀਂ 10++ ਜਾਂ - -10 ਦੀ ਵਰਤੋਂ ਕਰਾਂਗੇ ਤਾਂ ਇਹ ਗਲਤ ਸਟੇਟਮੈਂਟਸ ਮੰਨੀਆਂ ਜਾਣਗੀਆਂ ਜਿਵੇਂ ਕਿ ਪਹਿਲਾਂ ਹੀ ਦੱਸਿਆ ਜਾ ਚੁੱਕਿਆ ਹੈ ਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕਿਸੇ ਸਥਿਰ ਮੁੱਲ ਉਪਰ ਸਿੱਧੇ ਲਾਗੂ ਨਹੀਂ ਕਿਤਾ ਜਾ ਸਕਦਾ।

ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਦੋ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਤਰੀਕੇ ਇਸ ਗੱਲ ਉਪਰ ਨਿਰਭਰ ਕਰਦੇ ਹਨ ਕਿ ਆਪਰੇਟਰ ਨੂੰ ਆਪਰੇਂਡ ਤੋਂ ਬਾਅਦ ਲਿਖਿਆ ਗਿਆ ਹੈ ਜਾਂ ਪਹਿਲਾਂ। ਇਹ ਹੇਠ ਲਿਖੇ ਦੋ ਕਿਸਮਾਂ ਦੇ ਹੋ ਸਕਦੇ ਹਨ:

- ਪ੍ਰੀ-ਫਿਕਸ ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Prefix increment and decrement)
- ਪੋਸਟ-ਫਿਕਸ ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Postfix increment and decrement)

ਜੇਕਰ ਆਪਰੇਟਰ ਨੂੰ ਆਪਰੇਂਡ ਤੋਂ ਪਹਿਲਾਂ ਲਗਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਆਪਰੇਂਡ ਨੂੰ ਵਰਤਣ ਤੋਂ ਪਹਿਲਾਂ ਉਸਦਾ ਮੁੱਲ ਬਦਲਿਆ ਜਾਵੇਗਾ। ਇਸ ਨੂੰ ਪ੍ਰੀ-ਇੰਕਰੀਮੈਂਟ/ਡਿਕਰੀਮੈਂਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਪਰੰਤੂ, ਜੇਕਰ ਆਪਰੇਟਰ ਆਪਰੇਂਡ ਤੋਂ ਬਾਅਦ ਲਗਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਆਪਰੇਂਡ ਦਾ ਮੁੱਲ ਉਸਦੀ ਵਰਤੋਂ ਹੋਣ ਤੋਂ ਬਾਅਦ ਬਦਲਿਆ ਜਾਵੇਗਾ। ਇਸ ਨੂੰ ਪੋਸਟ ਇੰਕਰੀਮੈਂਟ/ਡਿਕਰੀਮੈਂਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।।

ਉਦਾਹਰਨ ਲਈ :

ਜੇਕਰ x ਦਾ ਮੁੱਲ ਸ਼ੁਰੂਆਤ ਵਿਚ 10 ਹੈ ਤਾ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਇਸਦਾ ਮੁੱਲ ਦੋ ਤਰੀਕਿਆਂ ਨਾਲ ਵਧਾਇਆ ਜਾ ਸਕਦਾ ਹੈ:

y = ++x;   (ਪ੍ਰੀ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦੇ ਮੁੱਲ ਵਿਚ ਵਾਧਾ ਕਰਕੇ 11 ਹੋ ਜਾਵੇਗਾ, ਉਸ ਤੋਂ ਬਾਅਦ x ਦਾ ਵਧਿਆ ਹੋਇਆ ਮੁੱਲ ਵੇਰੀਏਬਲ y ਵਿਚ ਸਟੋਰ ਕਰ ਦਿਤਾ ਜਾਵੇਗਾ, ਭਾਵ y ਦਾ ਮੁੱਲ ਵੀ 11 ਹੋ ਜਾਵੇਗਾ, (i.e. x =11 ਅਤੇ y =11)

y = x++;   (ਪੋਸਟ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦਾ ਮੁੱਲ ਵੇਰੀਏਬਲ y ਵਿਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ (ਭਾਵ y ਦਾ ਮੁੱਲ 10 ਹੋ ਜਾਵੇਗਾ) ਇਸ ਤੋਂ ਬਾਅਦ x ਦੇ ਮੁੱਲ ਵਿਚ ਵਾਧਾ ਹੋਣ ਤੋਂ ਬਾਅਦ 11 ਹੋ ਜਾਵੇਗਾ, (i.e. y = 10 ਅਤੇ x = 11)

ਇਸੇ ਤਰ੍ਹਾਂ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰ ਨੂੰ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ

y = --x;   (ਪ੍ਰੀ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦੇ ਮੁੱਲ ਵਿੱਚੋਂ 1 ਘਟਾਇਆ ਜਾਵੇਗਾ।ਅਤੇ x ਦਾ ਮੁੱਲ 9 ਹੋ ਜਾਵੇਗਾ ਅਤੇ ਬਆਦ ਵਿੱਚ ਇਹ x ਦਾ 9 ਮੁੱਲ y ਮੁੱਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ ਅਤੇ y ਦਾ ਮੁੱਲ ਵੀ 9 ਹੋ ਜਾਵੇਗਾ (i.e. x = 9 and y=9)

y = x--;   (ਪੋਸਟ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦਾ ਮੁੱਲ ਵੇਰੀਏਬਲ y ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ ਇਸ ਤੋਂ ਬਾਅਦ x ਦੇ ਮੁੱਲ ਵਿੱਚੋਂ ਇੱਕ ਨੰਬਰ ਘਟਾਇਆ ਜਾਵੇਗਾ ਅਤੇ x ਦਾ ਮੁੱਲ 9 ਹੋ ਜਾਵੇਗਾ, (i.e. x = 9 ਅਤੇ y = 9)

5.3.7 ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ Conditional Operator (? :)

ਇਹ ਇੱਕ ਟਰਨਰੀ ਆਪਰੇਟਰ (ternary) ਹੈ। ਜਾਵਾ ਵਿੱਚ ਕੇਵਲ ਇੱਕ ਹੀ ਟਰਨਰੀ ਓਪਰੇਟਰ ਹੈ। ਇਸ ਆਪਰੇਟਰ ਨੂੰ ਆਪਣਾ ਕੰਮ ਕਰਨ ਲਈ ਤਿੰਨ ਆਪਰੇਂਡਾਂ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਟਰਨਰੀ ਓਪਰੇਟਰ ਨੂੰ ਦਰਸਾਉਣ ਲਈ "? :" ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਸ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

exp1?exp2:exp3;

ਇੱਥੇ exp1 ਇੱਕ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੈ ਜੋ true ਜਾਂ false ਨਤੀਜਾ ਪੈਦਾ ਕਰਦਾ ਹੈ।ਜੇਕਰ ਐਕਸਪ੍ਰੈਸ਼ਨ exp1 ਦਾ ਨਤੀਜਾ true ਹੋਵੇਗਾ ਤਾਂ ਐਕਸਪ੍ਰੈਸ਼ਨ exp2 ਆਪਣਾ ਕੰਮ ਕਰੇਗੀ ਨਹੀਂ ਤਾਂ ਐਕਸਪ੍ਰੈਸ਼ਨ exp3 ਆਪਣਾ ਕੰਮ ਕਰੇਗੀ। ਉਦਾਹਰਣ ਲਈ:

```
a=5;
b=10;
c=a>b ? a : b;
```

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਐਕਸਪ੍ਰੈਸ਼ਨ (Operand1/exp1) a>b ਦਾ ਨਤੀਜਾ false ਹੈ, ਇਸ ਲਈ : ਤੋਂ ਬਾਅਦ ਵਾਲੀ ਐਕਸਪ੍ਰੈਸ਼ਨ (Operand3/exp3) ਵੇਰੀਏਬਲ b ਦਾ ਮੁੱਲ ਵੇਰੀਏਬਲ c ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ। ਐਕਸਪ੍ਰੈਸ਼ਨ (Oprand2/exp2) ਵੇਰੀਏਬਲ a ਕੋਈ ਕੰਮ ਨਹੀਂ ਕਰੇਗਾ ਕਿਉਂਕਿ ਇਹ ਆਪਣਾ ਕੰਮ ਤਾਂ ਹੀ ਕਰੇਗਾ ਜੇਕਰ exp1 ਦਾ ਨਤੀਜਾ (True) ਹੋਵੇਗਾ।

ਪ੍ਰੋਗਰਾਮ 5.4 ਜਾਵਾ ਵਿਚ ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਸੰਬਧੀ ਪ੍ਰੋਗਰਾਮ ਦੀ ਉਦਾਹਰਣ

Test4.java

```java
class Test4
{
    public static void main(String args[])
    {
        int num1=15, num2=10;
        int result;
        System.out.println("Value of num1 is "+num1);
        System.out.println("Value of num2 is "+num2);
        result = num1>num2 ? num1 : num2;
        System.out.println("Largest Number is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5.4 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ:

```
D:\Java Programs>javac Test4.java

D:\Java Programs>javac Test4.java

D:\Java Programs>java Test4
Value of num1 is 15
Value of num2 is 10
Largest Number is 15

D:\Java Programs>
```

5.3.8 ਸਪੇਸ਼ਲ ਆਪਰੇਟਰਜ਼ (Special Operators) :

JAVA ਕੁੱਝ ਵਿਸ਼ੇਸ਼ ਆਪਰੇਟਰਜ਼ ਵੀ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ: instance of ਆਪਰੇਟਰ ਅਤੇ ਮੈਂਬਰ ਸਿਲੈਕਸ਼ਨ (Member Selection) ਆਪਰੇਟਰ:

❖ **instranceof ਆਪਰੇਟਰ:** ਇਸ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਇਹ ਜਾਂਚ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਦਿੱਤਾ ਗਿਆ ਆਬਜੈਕਟ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਕਲਾਸ ਨਾਲ ਸਬੰਧਤ ਹੈ ਜਾਂ ਨਹੀਂ। ਇਹ ਇਸ ਗੱਲ 'ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ ਕਿ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਖੱਬੇ ਪਾਸੇ ਵਾਲਾ ਆਬਜੈਕਟ ਸੱਜੇ ਪਾਸੇ ਵਾਲੀ ਕਲਾਸ ਦਾ ਇੰਸਟਾਂਸ (Instance) ਹੈ ਜਾਂ ਨਹੀਂ। ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੀ ਜਾਂਚ ਤੋਂ ਬਾਅਦ ਇਹ true ਜਾਂ false ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ:

mango **instanceof** fruit

ਇਹ ਸਟੇਟਮੈਂਟ true ਵਾਪਸ ਕਰੇਗਾ ਜੇਕਰ mango ਆਬਜੈਕਟ fruit ਕਲਾਸ ਨਾਲ ਸਬੰਧਤ ਆਬਜੈਕਟ ਹੋਵੇਗਾ, ਨਹੀਂ ਤਾਂ ਇਹ false ਵਾਪਸ ਕਰੇਗਾ। ।

- **ਮੈਂਬਰ ਸਿਲੈਕਸ਼ਨ ਆਪਰੇਟਰ (Member Selection Operator):** ਇਸਨੂੰ ਡਾਟ (.) ਆਪਰੇਟਰ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਿਸੇ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਦੁਆਰਾ ਉਸ ਕਲਾਸ ਦੇ ਵੇਰੀਏਬਲਾਂ ਅਤੇ ਮੈਥਡਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

```
Object.variable;      //accessing variable of a class through object
Object.method();      //accessing method of a class through object
```

5.4 ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (EXPRESSIONS)

ਐਕਸਪ੍ਰੈਸ਼ਨ ਗਣਿਤ ਵਿਚ ਇਕ ਫਾਰਮੂਲੇ ਦੀ ਤਰ੍ਹਾਂ ਹੁੰਦੀ ਹੈ। ਇਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਓਪਰੈਂਡਜ਼ ਦਾ ਕੋਈ ਵੀ ਯੋਗ ਸੁਮੇਲ (valid combination) ਹੋ ਸਕਦਾ ਹੈ। ਇਕ ਯੋਗ ਸੁਮੇਲ ਅਜਿਹਾ ਸੁਮੇਲ ਹੁੰਦਾ ਹੈ ਜੋ JAVA ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ ਨਿਯਮਾਂ ਦੀ ਪੁਸ਼ਟੀ ਕਰਦਾ ਹੈ। ਇਕ ਯੋਗ ਐਕਸਪ੍ਰੈਸ਼ਨ ਨੂੰ well formed-expression ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਐਕਸਪੈਸ਼ਨ ਹਮੇਸ਼ਾ ਇਕੋ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਇਸ ਨਤੀਜੇ ਨੂੰ JAVA ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਐਕਸਪੈਸ਼ਨ ਇਕ ਸਿੰਗਲ ਮੁੱਲ ਜਿੰਨੀ ਸਾਧਾਰਣ ਵੀ ਹੋ ਸਕਦੀ ਹੈ ਅਤੇ ਇਕ ਵੱਡੀ ਕੈਲਕੁਲੇਸ਼ਨ ਜਿੰਨੀ ਗੁੰਝਲਦਾਰ ਵੀ। ਉਦਾਹਰਣ ਲਈ:

x= 2.9;

ਇਹ ਇਕ ਸਧਾਰਣ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੈ ਜਿਸ ਨੂੰ ਅਪਰੈਂਡਜ਼ x ਅਤੇ 2.9 ਨਾਲ ਵਰਤਿਆ ਗਿਆ ਹੈ।

x = 2.9 * y + 3.6 > z – (3.4 / z);

ਇਹ ਕੁੱਝ ਗੁੰਝਲਦਾਰ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੈ ਜੋ ਕਿ ਕਈ ਆਪਰੇਟਰਾ ਅਤੇ ਆਪਰੈਂਡਜ਼ ਤੋਂ ਮਿਲ ਕੇ ਬਣੀ ਹੋਈ ਹੈ। ਇਸ ਵਿਚ =, *, +, >, –/ ਆਪਰੇਟਰਜ਼ ਹਨ ਅਤੇ x, 2.9, y, 3.6, 34, ਆਪਰੈਂਡਜ਼ ਹਨ। ਹੇਠਾਂ ਦਿਤੀ ਉਦਾਹਰਣ ਵਿਚ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਅਪਰੈਂਡਜ਼ ਦਾ ਸੁਮੇਲ ਇਕ ਯੋਗ ਐਕਸਪੈਸ਼ਨ ਨਹੀਂ ਬਣਾ ਰਹੇ:

x + y = z;

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਭਾਵੇਂ ਅਸੀਂ ਯੋਗ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਆਪਰੈਂਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹਾਂ ਪ੍ਰੰਤੂ ਫਿਰ ਵੀ ਉਹਨਾਂ ਦਾ ਸੁਮੇਲ ਇਕ ਯੋਗ ਐਕਸਪ੍ਰੈਸ਼ਨ ਨਹੀਂ ਬਣਾ ਰਿਹਾ ਕਿਉਂਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਆਪਰੈਂਡਜ਼ ਨੂੰ JAVA ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ ਨਿਯਮਾਂ ਅਨੁਸਾਰ ਨਹੀਂ ਵਰਤਿਆ ਗਿਆ। ਇਸ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ z ਦਾ ਮੁੱਲ ਸਟੋਰ ਕਰਵਾਉਣ ਲਈ = ਆਪਰੇਟਰ ਦੇ ਖੱਬੇ ਪਾਸੇ ਇਕ ਯੋਗ ਮੈਮਰੀ ਲੋਕੇਸ਼ਨ (ਆਈਡੈਂਟੀਫਾਇਰ ਦਾ ਹੋਣਾ ਜਰੂਰੀ ਹੈ, ਜਦੋਂ ਕਿ ਉਪਰੋਕਤ ਉਦਾਹਰਣ ਵਿਚ ਖੱਬੇ ਪਾਸੇ x + y ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਗਈ ਹੈ ਜੋ ਕਿ ਇਕ ਯੋਗ ਆਈਡੈਂਟੀਫਾਇਰ ਨਹੀਂ ਹੋ ਸਕਦਾ ਕਿਉਂਕਿ ਇਕ ਯੋਗ ਆਈਡੈਂਟੀਫਾਇਰ ਵਿਚ ਅੰਡਰਸਕੋਰ ਤੋਂ ਇਲਾਵਾ ਹੋਰ ਕੋਈ ਵੀ ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹ ਨਹੀਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ (ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਪਿਛਲੇ ਪਾਠ ਵਿਚ ਪੜਿਆ ਹੈ)

ਜਾਵਾ ਭਾਸ਼ਾ ਦੀਆਂ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਨੂੰ ਹੇਠ ਲਿਖੀਆਂ ਦੋ ਸ਼੍ਰੇਣੀਆਂ ਵਿਚ ਵੰਡਿਆ ਜਾ ਸਦਕਾ ਹੈ:

5.4.1 ਨੂਮੇਰੀਕਲ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Numerical Expressions):

ਇਹਨਾਂ ਐਸਪੈਸ਼ਨਜ਼ ਦੀ ਵਰਤੋਂ ਲਾਜੀਕਲ ਅਤੇ ਕੰਡੀਸ਼ਨਲ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਹਮੇਸ਼ਾ ਦੋ ਸੰਭਵ ਮੁੱਲਾਂ ਵਿਚੋਂ ਇਕ ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੀਆਂ ਹਨ:

4 + 3

3.2 - 7.8

ਉਪਰੋਕਤ ਨੁਮੇਰੀਕਲ ਐਕਸਪੈਸ਼ਨਜ਼ ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਨੁਮੇਰੀਕਲ ਮੁੱਲ 7 ਅਤੇ –4.6 ਪ੍ਰਦਾਨ ਕਰਣਗੀਆਂ।

5.4.2 ਲਾਜੀਕਲ ਜਾਂ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Logical or Conditional Expressions):

ਇਹਨਾਂ ਐਸਪੈਸ਼ਨਜ਼ ਦੀ ਵਰਤੋਂ ਲਾਜੀਕਲ ਅਤੇ ਕੰਡੀਸ਼ਨਲ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਹਮੇਸ਼ਾ ਦੋ ਸੰਭਵ ਮੁੱਲਾਂ ਵਿਚੋਂ ਇਕ ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੀਆਂ ਹਨ:

14 > 6

15 <= 6

ਉਪਰੋਕਤ ਪਹਿਲੀ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਸਾਨੂੰ true ਨਤੀਜਾ ਦੇਵੇਗੀ ਜਦੋਂ ਕਿ ਦੂਸਰੀ ਐਕਸਪ੍ਰੈਸ਼ਨ false ਨਤੀਜਾ ਪ੍ਰਦਾਨ ਕਰੇਗੀ।

5.5 ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ/ਹਰਾਰਕੀ (PRECEDENCE/ HIERARCHY AND ASSOCIATIVITY OF OPERATORS)

ਉਹ ਕ੍ਰਮ ਜਾਂ ਤਰਜੀਹ (order or priority) ਜਿਸ ਵਿਚ ਕਿਸੇ ਦਿੱਤੀ ਗਈ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ (evaluation) ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉੱਚ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰ ਘੱਟ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਤੋਂ ਪਹਿਲਾਂ ਆਪਣਾ ਕੰਮ ਕਰਦੇ ਹਨ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਆਪਰੇਟਰਾਂ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕੰਮ ਜਿਸ ਕ੍ਰਮ ਵਿੱਚ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੈਂਡਾ ਉੱਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਭਾਗ (division) ਕਰਨ ਦਾ ਕੰਮ ਘਟਾਓ (subtraction) ਤੋਂ ਪਹਿਲਾਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਭਾਗ ਓਪਰੇਟਰ ਨੂੰ ਘਟਾਓ ਆਪਰੇਟਰ ਨਾਲੋਂ ਵੱਧ ਤਰਜੀਹ ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ। ਬਰੈਕਟਾਂ (Parentheses) ( ) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਸੀਂ ਆਪਰੇਟਰਾਂ ਦੇ ਕੰਮ ਕਰਨ ਦੀ ਤਰਜੀਹ ਨੂੰ ਬਦਲ ਸਕਦੇ ਹਾਂ। ਆਮ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਦੀ ਤਰਜੀਹ ਘਟਦੇ ਕ੍ਰਮ ਵਿੱਚ ਅਤੇ ਉਹਨਾਂ ਦੀ ਐਸੋਸੀਏਟੀਵਿਟੀ (Associativity) ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਹੇਠਾਂ ਟੇਬਲ ਵਿਚ ਦਿੱਤੀ ਗਈ ਹੈ:

ਲੜੀ ਨੰ:	ਆਮ ਵਰਤੇ ਜਾਂਦੇ ਆਪਰੇਟਰਜ਼	ਐਸੋਸੀਏਟੀਵਿਟੀ
1.	. ( ) [ ]	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
2.	- ++ - - ! ~ (type) casting	ਸੱਜੇ ਤੋਂ ਖੱਬੇ
3.	* / %	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
4.	+ -	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
5.	< <= > > = instanceof	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
6.	== !=	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
7.	&&	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
8.	||	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
9.	? :	ਸੱਜੇ ਤੋਂ ਖੱਬੇ
10.	= *= /= %- += -=	ਸੱਜੇ ਤੋਂ ਖੱਬੇ

ਹੇਠਾਂ ਦਿੱਤੀ ਉਦਾਹਰਨ ਇਹ ਦਰਸਾਉਂਦੀ ਹੈ ਕਿ ਓਪਰੇਟਰ ਦੀ ਤਰਜੀਹ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਰਥਮੈਟਿਕ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਦਾ ਮੁਲਾਂਕਣ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ:

a = 5 * 4 / 4 + 8 – 9 / 3; (* ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 20 / 4 + 8 – 9 / 3; (/ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 5 + 8 – 9 / 3;   (/ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 5 + 8 - 3;   (+ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 13 - 3;   (- ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 10;   ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਨਤੀਜਾ

ਉਹ ਕ੍ਰਮ ਜਿਸ ਵਿੱਚ ਇੱਕੋ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਐਸੋਸੀਏਟੀਵਿਟੀ (Associativity) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਜੋੜ ਅਤੇ ਘਟਾਓ ਆਪਰੇਟਰਾਂ ਦੀ ਤਰਜੀਹ ਇੱਕੋ ਜਿਹੀ ਹੁੰਦੀ ਹੈ। ਹਾਲਾਂਕਿ, ਜੋੜ ਅਤੇ ਘਟਾਓ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕ੍ਰਮ ਉਹਨਾਂ ਦੇ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਮੌਜੂਦ ਹੋਣ ਤੇ ਕ੍ਰਮ ਉਪਰ ਨਿਰਭਰ ਕਰਦਾ ਹੈ। ਕਿਸੇ ਓਪਰੇਟਰ ਦੀ ਐਸੋਸੀਏਟੀਵਿਟੀ ਜਾਂ ਤਾਂ ਖੱਬੇ ਤੋਂ ਸੱਜੇ ਜਾਂ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਹੋ ਸਕਦੀ ਹੈ। ਖੱਬੇ ਤੋਂ ਸੱਜੇ ਐਸੋਸੀਏਟੀਵਿਟੀ ਵਾਲੇ ਓਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਖੱਬੇ ਸਿਰੇ ਤੋਂ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਐਸੋਸੀਏਟੀਵਿਟੀ ਵਾਲੇ ਓਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਸੱਜੇ ਸਿਰੇ ਤੋਂ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ।

ਉਦਾਹਰਣ ਲਈ:

a = b = c = 8;

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੀ ਐਸੋਸੀਏਟੀਵਿਟੀ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਹੁੰਦੀ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੈ ਕਿ ਇਸ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ 8 ਮੁੱਲ c ਵੇਰੀਏਬਲ ਨੂੰ ਅਸਾਈਨ ਕੀਤਾ ਜਾਵੇਗਾ, ਫਿਰ c ਦਾ ਮੁੱਲ b ਨੂੰ ਅਸਾਈਨ ਕੀਤਾ ਜਾਵੇਗਾ, ਅਤੇ ਅੰਤ ਵਿੱਚ b ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਲ a ਨੂੰ ਅਸਾਈਨ ਕੀਤਾ ਜਾਵੇਗਾ। ਇਸ ਤਰ੍ਹਾਂ ਇਸ ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟ ਵਿਚ = ਆਪਰੇਟਰ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਵੱਲ ਜਾਂਦੇ ਹੋਏ ਆਪਣੇ ਕੰਮ ਪੂਰਾ ਕਰਨਗੇ।

5.6 ਟਾਈਪ ਕਨਵਰਜ਼ਨ (TYPE CONVERSION)

ਜਾਵਾ ਵਿਚ ਜਰੂਰਤ ਅਨੁਸਾਰ ਕਿਸੇ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਮੁੱਲ ਕਿਸੇ ਖਾਸ ਵੱਖਰੀ ਟਾਈਪ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਕਿਸੇ ਇਕ ਕਿਸਮ ਦੇ ਮੁੱਲ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਕਿਸਮ ਦੇ ਮੁੱਲ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਨੂੰ ਟਾਈਪ ਕਨਵਰਜ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਵਿਚ ਇਹ ਕਨਵਰਜ਼ਨ ਦੋ ਤਰੀਕਿਆਂ ਨਾਲ ਹੋ ਸਕਦੀ ਹੈ:

1. ਇੰਪਲੀਸਿਟ (ਪ੍ਰਤੱਖ) ਕਨਵਰਜ਼ਨ Implicit Conversion or Widening Conversion
2. ਐਕਸਪਲੀਸਿੱਟ (ਸਪਸ਼ਟ) ਕਨਵਰਜ਼ਨ Explicit Conversion or Narrowing Conversion

5.6.1 ਇੰਪਲੀਸਿਟ ਕਨਵਰਜ਼ਨ ਜਾਂ ਵਾਈਡਨਿੰਗ ਕਨਵਰਜ਼ਨ (Implicit Conversion or Widening Conversion):

ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਆਟੋਮੈਟਿਕ (automatic) ਹੁੰਦੀ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਲਈ ਅਸੀਂ ਅਸਾਈਨਮੈਂਟ (=) ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਦੀ ਵਰਤੋਂ ਉਸ ਸਮੇਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਛੋਟੀ ਡਾਟਾ ਟਾਈਪ ਵਾਲੇ ਆਪਰੈਂਡ ਨੂੰ ਵੱਡੀ ਡਾਟਾ ਟਾਈਪ ਵਿੱਚ ਬਦਲਿਆ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਡਾਟਾ ਕਨਵਰਜ਼ਨ ਵਿਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਦੋਵੇਂ ਡਾਟਾ ਆਈਟਮਾਂ ਅਨੁਕੂਲ (compatible) ਹੋਣ ਅਤੇ ਟਾਰਗੇਟ ਵੇਰੀਏਬਲ ਦੀ ਡਾਟਾ ਟਾਈਪ ਸੋਰਸ ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਵੱਡੀ ਹੋਵੇ, ਤਾਂ ਜਾਵਾ ਆਪਣੇ ਆਪ ਸੋਰਸ ਟਾਈਪ ਨੂੰ ਟਾਰਗੇਟ ਟਾਈਪ ਵਿੱਚ ਬਦਲ ਦਿੰਦਾ ਹੈ। ਇਸ ਕਿਸਮ ਦੇ ਕਨਵਰਜ਼ਨ ਵਿੱਚ ਜਾਣਕਾਰੀ ਦਾ ਕੋਈ ਨੁਕਸਾਨ ਨਹੀਂ (no loss of information) ਹੁੰਦਾ। ਉਦਾਹਰਨ ਲਈ: int ਡਾਟਾ ਟਾਈਪ ਦਾ ਮੁੱਲ (double) ਡਾਟਾ ਟਾਈਪ ਦੇ ਇੱਕ ਵੇਰੀਏਬਲ ਨੂੰ ਦਿੱਤਾ ਜਾ ਸਕਦਾ ਹੈ ਕਿਉਂਕਿ (double) ਟਾਈਪ int ਟਾਈਪ ਤੋਂ ਵੱਡੀ ਹੁੰਦੀ ਹੈ। ਆਟੋਮੈਟਿਕ ਕਨਵਰਜ਼ਨ ਲਈ ਅੱਗੇ ਦਿੱਤੀ ਉਦਾਹਰਣ ਦੇਖੋ:

```
double n;
n = 5;
```

ਇਸ ਉਦਾਹਰਨ ਵਿੱਚ ਇੰਪਲੀਸਿੱਟ ਕਨਵਰਜ਼ਨ ਹੋਵੇਗਾ, ਕਿਉਂਕਿ int ਟਾਈਪ ਦਾ ਮੁੱਲ (5) ਆਪਣੇ ਆਪ double ਟਾਈਪ ਦੇ ਮੁੱਲ ਵਿੱਚ ਬਦਲ ਜਾਵੇਗਾ ਤਾਂ ਜੋ ਇਸਨੂੰ double ਟਾਈਪ ਦੇ ਵੇਰੀਏਬਲ n ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾ ਸਕੇ, ਭਾਵ ਵੇਰੀਏਬਲ n ਦਾ ਮੁੱਲ 5.0d

ਬਣ ਜਾਵੇਗਾ।

5.6.2 ਐਕਸਪਲੀਸਿੱਟ ਕਨਵਰਜ਼ਨ ਜਾਂ ਨੈਰੋਇੰਗ ਕਨਵਰਜ਼ਨ ਜਾਂ ਟਾਈਪ ਕਾਸਟਿੰਗ (Explicit Conversion or Narrowing Conversion or Type Casting):

ਜੇਕਰ ਟਾਰਗੇਟ ਟਾਈਪ ਸੋਰਸ ਟਾਈਪ ਤੋਂ ਛੋਟਾ ਹੈ, ਤਾਂ ਕਨਵਰਜ਼ਨ ਆਪਣੇ ਆਪ ਨਹੀਂ ਹੁੰਦੀ। ਉਦਾਹਰਨ ਲਈ, (double) ਟਾਈਪ ਦਾ ਮੁੱਲ int ਟਾਈਪ ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਅਜਿਹੇ ਕਨਵਰਜ਼ਨ ਲਈ Java ਇੱਕ ਤਕਨੀਕ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਸਨੂੰ ਟਾਈਪ ਕਾਸਟਿੰਗ (Type Casting) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।ਇਹ ਜ਼ਬਰਦਸਤੀ (forceful) ਕਨਵਰਜ਼ਨ ਹੁੰਦੀ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਲਈ ਅਸੀਂ (caste) ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਸ ਕਿਸਮ ਦੇ ਕਨਵਰਜ਼ਨ ਵਿੱਚ ਕਨਵਰਜ਼ਨ ਪੂਰੀ ਹੋਣ ਤੋਂ ਬਾਅਦ ਜਾਣਕਾਰੀ ਦਾ ਕੋਈ ਨੁਕਸਾਨ (loss of information) ਹੋ ਵੀ ਸਕਦਾ ਹੈ ਜਾਂ ਨਹੀਂ ਵੀ।

ਇਸ ਕਿਸਮ ਦੀ ਕਾਸਟਿੰਗ ਲਈ ਸਿੰਟੈਕਸ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

(**data type**) variable ਜਾਂ expression

ਉਹ ਡਾਟਾ ਟਾਈਪ ਜਿਸ ਵਿੱਚ ਕਨਵਰਜ਼ਨ ਕੀਤੀ ਜਾਣੀ ਹੈ, ਉਸਨੂੰ ਬਰੈਕਟਾਂ ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਕਨਵਰਟ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਸੋਰਸ ਮੁੱਲ ਨੂੰ ਬਰੈਕਟਾਂ ਤੋਂ ਬਾਅਦ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਟਾਈਪ ਕਾਸਟਿੰਗ ਦੀ ਉਦਾਹਰਣ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

double n=45.5;
int a=(**int**) n;

ਇਸ ਉਦਾਹਰਨ ਵਿੱਚ (double) ਡਾਟਾ ਟਾਈਪ ਨੂੰ int ਡਾਟਾਟਾਈਪ ਵਿੱਚ ਬਦਲਣ ਲਈ ਟਾਈਪ ਕਾਸਟਿੰਗ ਕੀਤੀ ਗਈ ਹੈ। ਇਸ ਵਿਚ int ਟਾਈਪ ਵੇਰੀਏਬਲ 'a' ਦਾ ਮੁ... ਵੇਗਾ ਅਤੇ .5 ਖਤਮ ਕਰ ਦਿਤਾ ਜਾਵੇਗਾ, ਭਾਵ ਜਾਣਕਾਰੀ ਦਾ ਨੁਕਸਾਨ ਹੋਵੇਗਾ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਡਾਟਾ ਟਾਈਪਸ ਇੱਕ ਖਾਸ ਕਿਸਮ, ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ ਅਤੇ ਡਾਟਾ ਉਪਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ ॥
2. ਪ੍ਰਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਬਿਲਟ-ਇਨ (built-in) ਡਾਟਾ ਟਾਈਪਸ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
3. JAVA ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ: byte, short, int ਅਤੇ long ਟਾਈਪਸ।
4. ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਡਾਟਾਟਾਈਪ ਦੀ ਵਰਤੋਂ ਰੀਅਲ ਨੰਬਰਾਂ ਜਿਵੇਂ ਕਿ: 3.14, 74.5884569, +29.05, -524836.458 ਆਦਿ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
5. ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਸਿੰਗਲ ਕੋਟਸ (') ਵਿੱਚ ਬੰਦ ਇੱਕ ਸਿੰਗਲ ਯੂਨੀਕੋਡ ਅੱਖਰ (single Unicode character enclosed in single quotes) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
6. boolean ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲਾਂ ਵਿਚ ਬੂਲੀਅਨ ਮੁੱਲਾਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਡਾਟਾ ਟਾਈਪ ਸਿਰਫ ਦੋ ਮੁੱਲਾਂ ਨੂੰ ਸਵੀਕਾਰ ਕਰਦਾ ਹੈ: **true** ਜਾਂ **false** ਮੁੱਲ।
7. ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਯੂਜ਼ਰ ਦੁਆਰਾ ਪਰਿਭਾਸ਼ਿਤ (user-defined) ਡਾਟਾ ਟਾਈਪਸ ਜਾਂ ਰੈਫਰੈਂਸ (reference) ਡਾਟਾ ਟਾਈਪਸ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
8. ਵੇਰੀਏਬਲ ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਹੁੰਦਾ ਹੈ।ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ (memory location) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।
9. ਅਸੀਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਸਮੇਂ ਉਸ ਨੂੰ ਇੱਕ ਮੁੱਲ ਵੀ ਅਸਾਈਨ (assign) ਕਰ ਸਕਦੇ ਹਾਂ, ਇਸ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਵੇਰੀਏਬਲ ਇਨੀਸ਼ੀਅਲਾਈਜੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

10. ਆਪਰੈਂਡਜ਼ ਉਹ ਡਾਟਾ ਆਈਟਮਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜਿਨ੍ਹਾਂ ਉਪਰ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰ ਸਕਦੇ ਹੁੰਦੇ ਹਨ।

11. ਆਪਰੇਟਰਾਂ ਅਤੇ ਆਪਰੈਂਡਾਂ ਦੇ ਇੱਕ ਵੇਲੀਡ ਸਮੂਹ (vaid combination) ਨੂੰ ਐਸਕਪ੍ਰੈਸ਼ਨ/ਸਮੀਕਰਨ (Expression) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

12. ਅਰਿਥਮੈਟਿਕ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਅਰਿਥਮੈਟਿਕ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕਿਤੀ ਜਾਂਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ: ਜੋੜ, ਘਟਾਓ, ਗੁਣਾ, ਭਾਗ ਆਦਿ।

13. ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਤੁਲਨਾਤਮਕ (comparison) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਆਪਰੈਂਡਜ਼ ਵਿਚਕਾਰ ਰਿਲੇਸ਼ਨਸ਼ਿਪ ਟੈਸਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

14. ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਬੂਲੀਅਨ (Boolean) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

15. ਸ਼ਾਰਟਹੈਂਡ ਆਪਰੇਟਰਜ਼ ਸੈਲਫ-ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟਜ਼ ਵਿਚ ਲਾਭਦਾਇਕ ਹੁੰਦੇ ਹਨ।

16. ਇਨਕਰੀਮੈਂਟ (++) ਅਤੇ ਡਿਕਰੀਮੈਂਟ (--) ਆਪਰੇਟਰ ਯੂਨਰੀ ਆਪਰੇਟਰ ਹਨ।

17. ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ ਇੱਕ ਟਰਨਰੀ ਆਪਰੇਟਰ ਹੈ।

18. ਆਪਰੇਟਰਾਂ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕ੍ਰਮ ਜਿਸ ਵਿੱਚ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੈਂਡਾ ਉੱਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

19. ਉਹ ਕਮ ਜਿਸ ਵਿੱਚ ਇੱਕੋ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਐਸੋਸੀਏਟੀਵਿਟੀ (Associativity) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

20. ਜਦੋਂ ਕਿਸੇ ਇਕ ਕਿਸਮ ਦੇ ਮੁੱਲ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਕਿਸਮ ਦੇ ਮੁੱਲ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਨੂੰ ਟਾਈਪ ਕਨਵਰਜ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਅਭਿਆਸ

ਪ੍ਰ:1 ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ

1. ....ਇੱਕ ਖਾਸ ਕਿਸਮ, ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ ਅਤੇ ਡਾਟਾ ਉਪਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ।

ੳ. ਵੇਰੀਏਬਲ (Variable) ਅ. ਐਕਸਪ੍ਰੈਸ਼ਨ (Expression)
ੲ. ਆਪਰੈਂਡ (Operand) ਸ. ਡਾਟਾ ਟਾਈਪ (Data Type)

II. ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਡਾਟਾਟਾਈਪ ਦੀ ਵਰਤੋਂ .... ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ੳ. ਇੰਟੀਜਰਜ਼ (Integers) ਅ. ਕਰੈਕਟਰ (Character)
ੲ. ਰੀਅਲ ਨੰਬਰਜ਼ (Real number) ਸ. ਬੂਲੀਅਨ (Boolen)

III. ....ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ (memory location) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

ੳ. ਵੇਰੀਏਬਲ (Variable) ਅ. ਲਿਟਰਲ (Literal)
ੲ. ਇੰਟੀਜਰ (Integer) ਸ. ਆਪਰੇਟਰ (Operator)

IV. ....ਉਹ ਡਾਟਾ ਆਈਟਮਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜਿਨ੍ਹਾਂ ਉਪਰ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰ ਸਕਦੇ ਹੁੰਦੇ ਹਨ।

ੳ. ਆਪਰੈਂਡਜ਼ (Operands) ਅ. ਲਿਟਰਲ (Literal)
ੲ. ਡਾਟਾ ਟਾਈਪ (Data Type) ਸ. ਆਪਰੇਟਰ (Operator)

V. ....ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਆਪਰੈਂਡਜ਼ ਵਿਚਕਾਰ ਰਿਲੇਸ਼ਨਸ਼ਿਪ ਟੈਸਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ੳ. ਅਰਿਥਮੈਟਿਕ (Arithmetic) ਅ. ਯੂਨਰੀ (Unary)

ੲ. ਰਿਲੇਸ਼ਨਲ (Relational) ਸ. ਟਰਨਰੀ (Ternary)

VI . ਇਨਕਰੀਮੈਂਟ (++) ਅਤੇ ਡਿਕਰੀਮੈਂਟ (--) ਆਪਰੇਟਰ .... ਆਪਰੇਟਰ ਹਨ।

ੳ. ਯੂਨਰੀ (Unary) ਅ. ਬਾਇਨਰੀ (Binary)

ੲ. ਟਰਨਰੀ (Ternary) ਸ. ਇਹਨਾਂ ਵਿਚੋਂ ਕੋਈ ਨਹੀਂ

VII . ਜਦੋਂ ਕਿਸੇ ਇਕ ਕਿਸਮ ਦੇ ਮੁੱਲ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਕਿਸਮ ਦੇ ਮੁੱਲ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਡਾਟਾ ਕਨਵਰਜ਼ਨ (Data Conversion) ਅ. ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion)

ੲ. ਵੈਲਿਯੂ ਕਨਵਰਜ਼ਨ (Value Conversion) ਸ. ਆਪਰੇਟਰ ਕਨਵਰਜ਼ਨ (Operator Conversion)

ਪ੍ਰ:2 ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ।

I. JAVA ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ int ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ: byte, short, int ਅਤੇ .... ਟਾਈਪਸ।

II. ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ......................... ਵਿੱਚ ਬੰਦ ਇੱਕ ਸਿੰਗਲ ਯੂਨੀਕੋਡ ਅੱਖਰ (single Unicode character) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

III. ਆਪਰੇਟਰਾਂ ਅਤੇ ਅਪਰੈਂਡਾਂ ਦੇ ਇੱਕ ਵੇਲੀਡ ਸਮੂਹ (vaid combination) ਨੂੰ .... ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

IV. ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ .... ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

V. ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ ਇੱਕ .... ਆਪਰੇਟਰ ਹੈ।

VI. ਆਪਰੇਟਰਾਂ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕ੍ਰਮ ਜਿਸ ਵਿੱਚ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੈਂਡਾ ਉੱਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਪ੍ਰ:3 ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

I. ਡਾਟਾ ਟਾਈਪਸ (Data Types) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

ii. ਜਾਵਾ ਵਿਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਵੱਖ-ਵੱਖ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Primitive Data Type) ਦੇ ਨਾਂ ਲਿਖੋ।

iii. ਵੇਰੀਏਬਲਜ਼ (Variables) ਕੀ ਹੁੰਦੇ ਹਨ ?

iv. ਆਪਰੈਂਡਜ਼ (Operands) ਕੀ ਹੁੰਦੇ ਹਨ ?

v. ਜਾਵਾ ਦੇ ਅਰਿੱਥਮੈਟਿੱਕ ਆਪਰੇਟਰਜ਼ (Arithmetic Operators) ਬਾਰੇ ਲਿਖੋ।

vi. ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ (assigment Operators) ਕੀ ਹੁੰਦੇ ਹਨ ?

vii. ਤੁਸੀਂ ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Increment and Decrement Operators) ਆਪਰੇਟਰਜ਼ ਬਾਰੇ ਕੀ ਜਾਣਦ ਹੋ ?

viii.ਜਾਵਾ ਵਿਚ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ?

ix. ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion) ਕੀ ਹੁੰਦੀ ਹੈ ?

ਪ੍ਰ:4 ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

i. ਐਕਸਪ੍ਰੈਸ਼ਨ (Expression) ਕੀ ਹੁੰਦੀ ਹੈ ? ਜਾਵਾ ਵਿਚ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਐਕਸਪ੍ਰੈਸ਼ਨਾਂ ਦਾ ਵਰਨਣ ਕਰੋ।

ii. ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ (Relational Operators) ਕੀ ਹੁੰਦੇ ਹਨ ? ਢੁੱਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਵਰਨਣ ਕਰੋ।

iii. ਲਾਜੀਕਲ ਆਪਰੇਟਰਾਂ (Logical Operators) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ ? ਢੁੱਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਵਰਨਣ ਕਰੋ।

iv. ਜਾਵਾ ਵਿਚ ਇਕ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰੋ ਜੋ ਜਾਵਾ ਦੇ ਅਰਿਥਮੈਟਿਕ ਆਰਪੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੋਵੇ।

v. ਅਸੀਂ ਜਾਵਾ ਵਿਚ ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ (Conditional Operator) ਦੀ ਵਰਤੋਂ ਕਿਸ ਤਰ੍ਹਾਂ ਕਰਾਂਗੇ ? ਢੁਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਵਰਨਣ ਕਰੋ। ।

vi. ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion) ਕੀ ਹੈ ? ਟਾਈਪ ਕਨਵਰਜ਼ਨ ਦੀਆਂ ਵੱਖ ਵੱਖ ਕਿਸਮਾਂ ਦਾ ਵਰਨਣ ਕਰੋ।