***Oparators***

1. **int x = 0, y = 0 , z = 0 ;**

**x = (++x + y-- ) \* z++;**

**What will be the value of "x" after execution ?**

A. -2

B. -1

C. 0

D. 1

E. 2

**Answer: Option C**

1. **int ++a = 100 ;**

**System.out.println( ++a ) ;**

**What will be the output of the above fraction of code ?**

A. 100

B. Displays error as ++a is not enclosed in double quotes in println statement

C. Compiler displays error as ++a is not a valid identifier

D. None of these

**Answer: Option C**

**3. What is the output of the following program ?**

**class Numbers{**

**public static void main(String args[]){**

**int a=20, b=10;**

**if((a < b) && (b++ < 25)){**

**System.out.println("This is any language logic");**

**}**

**System.out.println(b);**

**}**

**}**

A. 12

B. 11

C. 10

D. Compilation Error

**Answer: Option C**

**4 . Select from among the following character escape code which is not available in Java.**

A. t

B. r

C. a

D. \

E. v

**Answer: Option C**

**5. What will be the output?**

**if(1 + 1 + 1 + 1 + 1 == 5){**

**System.out.print("TRUE");**

**}**

**else{**

**System.out.print("FLASE");**

**}**

**A. TRUE**

**B. FALSE**

**C. Compiler Error**

**D. None of these**

**Answer: Option A**

**6. Which of the following is the correct expression that evaluates to true if the number x is between 1 and 100 or the number is negative?**

A. 1 < x < 100 || x < 0

B. ((x < 100) && (x > 1)) || (x < 0)

C. ((x < 100) && (x > 1)) && (x < 0)

D. (1 > x > 100) || (x < 0)

**Answer: Option B**

**7.** **public class Test{**

**public static void main(String args[]){**

**System.out.print(""=="");**

**System.out.print(" ");**

**System.out.print("A"=="A");**

**System.out.print(" ");**

**System.out.print("a==A");**

**}**

**}**

A. "==" A"=="A a==A

B. true true false

C. true true a==A

D. Compilation Fails

E. None of the above

**Answer: Option C**

**8.** **What will be the output?**

**public class Test{**

**public static void main(String args[]){**

**int a = 42;**

**double b = 42.25;**

**System.out.print((a%10)+" "+(b%10));**

**}**

**}**

A. 42 42.5

B. 2 2.5

C. 4.2 4.225

D. 2 4.225

E. Compilation Error

**Answer: Option B**

Solution:

The modulus operator, % returns the remainder of a division operation. It can be applied on floating-point types as well as integer types. (This differs from C/C++, in which the % can only be applied on integer types.)

**9.** **What will be the output after compiling and running following code?**

**1. public class Test{**

**2. public static void main(String... args){**

**3. int x =5;**

**4. x \*= 3 + 7;**

**5. System.out.println(x);**

**6. }**

**7. }**

A. 22

B. 50

C. 10

D. Compilation fails with an error at line 4

E. None of these

**Answer: Option B**

Solution:

x \*= 3 + 7; is same as x = x \* (3 +7) = 5 \* (10) = 50 because expression on the right side is always placed inside parentheses.

**10.** **public class Test{**

**public static void main(String... args){**

**int a=5 , b=6, c=7;**

**System.out.println("Value is "+ b + c);**

**System.out.println(a + b + c);**

**System.out.println("String " + (b+c));**

**}**

**}**

A. Value is 67 18 String 13

B. Value is 13 18 String 13

C. Value is 13 18 String

D. Compilation fails

E. None of these

**Answer: Option A**

Solution:

If the left hand operand is not a String then + operator treat as plus BUT if the left hand operand is a String then + perform String concatenation.

**11.** **What will be the output for the below code ?**

**class A{**

**public void printValue(){**

**System.out.println("A");**

**}**

**}**

**class B extends A{**

**public void printValue(){**

**System.out.println("B");**

**}**

**}**

**1. public class Test{**

**2. public static void main(String... args){**

**3. A b = new B();**

**4. newValue(b);**

**5. }**

**6. public static void newValue(A a){**

**7. if(a instanceof B){**

**8. ((B)a).printValue();**

**9. }**

**10. }**

**11. }**

A. A

B. B

C. Compilation fails with an error at line 4

D. Compilation fails with an error at line 8

E. None of these

**Answer: Option B**

Solution:

Instanceof operator is used for object reference variables to check whether an object is of a particular type. In newValue(b); b is instance of B so it works properly.

**12.** **public class Test{**

**static int i = 5;**

**public static void main(String... args){**

**System.out.println(i++);**

**System.out.println(i);**

**System.out.println(++i);**

**System.out.println(++i+i++);**

**}**

**}**

A. 6 6 6 16

B. 6 7 6 16

C. 5 6 7 16

D. 5 6 6 16

E. None of these

**Answer: Option C**

Solution:

i++ : print value then increment (postfix - increment happens after the value of the variable is used) ++i : increment the print (prefix - increment happens before the value of the variable is used).