Internals - 02

OOC (18CS45)

Shahul Hameed. S
1FN18CS097
CSE 'A' Sec

1a.

```
Public class Add
{
Public static void main (String[] args)
{
int [] list = {40, 60, 10, 20, 80};
int sum = 0
for (int element : list)
{
sum = sum + element;
}
System.out.println("The sum of first 5 elements is
:");
System.out.println(sum);
}
}
```

1b. (i) Remainder operator: Returns the remainder of the division.

Ex:

```
public class Rem
{
Public static void main(String[] args)
{
int a = 40, b = 2;
System.out.println(b%a);
}
}
```

(ii) Shift Right zero fill operator (>>>) :-

The left operand value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.

Ex:-

```
public class Shift
{
  Public class void main (String[] args)
  {
    int a = 40;
    System.out.println(a>>>2);
  }
}
```

(iii) (&&) Logical AND :- The logical && operator doesn't check second condition is false. It checks second condition only if first condition is true.

```
Public class And
{
  Public static void main (String[]args)
  {
    int a=50,b=50,c=20;
    System.out.println(a>b && c>a);
    System.out.println(a>b && b>c);
  }
}
```

(iv) Not operator (!) The bitwise ! operator always checks the conditions whether first condition is true or false

Ex:-

```
public class not
{
 Public static void main (String [] args)
 {
  int a=2,b=4, c=8;
  System.out.println (a>b!a*c);
  System.out.println(a>b!a+b<c);
  System.out.println(a);
 }
}
```

3a. A reference variable of a superclass can be assigned a reference to any subclass.

Ex:

```
class Bike
 {
 Bike()
    {
     System.out.println("Bike is created");
    }
 }

class Honda extends Bike
 {
  void run()
     {
     System.out.println("Running Safely");
     }
 }
```

```
class Test
{
    public static void main (String args[])
    {
        Bike B = new Honda();
        B.run();
    }
}
```

Here in the above example super class variable Bike extends sub class variable Honda / Peppers subclass.

3b.

Overloading	Overriding
1. Method overloading is used to increase the readability.	1. Overriding is used to provide the specific implementation of the method that is already provided by its super class.
2. Method overloading is performed within class.	2. Overriding occurs in two classes that have IS-A relationship.
3. In case of method overloading, parameter must be different.	3. Here parameter must be same.
4. It is an example of compile time polymorphism.	4. It is the example of run time polymorphism
5. In java overloading can't be performed by changing return type of the method only, Return type can be same or different. But must have change in parameter.	5. Return type must be same or covariant in method overridi

Ex:- Overloading:

```
class overload
{
static int add (int a, int b)
        {
        return a+b;
        }
static int add (int a, int b, int c)
        {
        return a+b+c;
        }
public static void main (String args[])
{
System.out.println (add(1,5));
System.out.println (add(1,2,3));
}
}
```

Ex:- Overriding:

```
class Animal {
    void eat()
        {
        System.out.println ("Eating");
        }
    class Dog extends Animal
    {
      void eat(){System.out.println ("Dog is Eating");}
    }
class override {
    public static void main (String args[])
    { Dog D = new Dog();
      Animal A = new Animal();
      D.eat();
      A.eat();
    }
}
```