Name: Shah fahad

Id: 16172

Program: BS Software

Section : (A)

Question 1:

a)

Access Modifiers:

Access Modifiers is use for access control, when we use class in a program so inside the class members, variables and methods of the class can specify with the help of access modifiers. Access modifiers are also called visibility modifiers.

Access Modifiers use on 2 levels:

i) At the Top level
Like class Level access modifiers

ii) At the Member Level
Like Member Level access modifiers

Private Access Modifiers:

In the above example we inherit class B with class A using keyword "extends" so in the class B we access all the variable of class A because here is no access modifiers use.

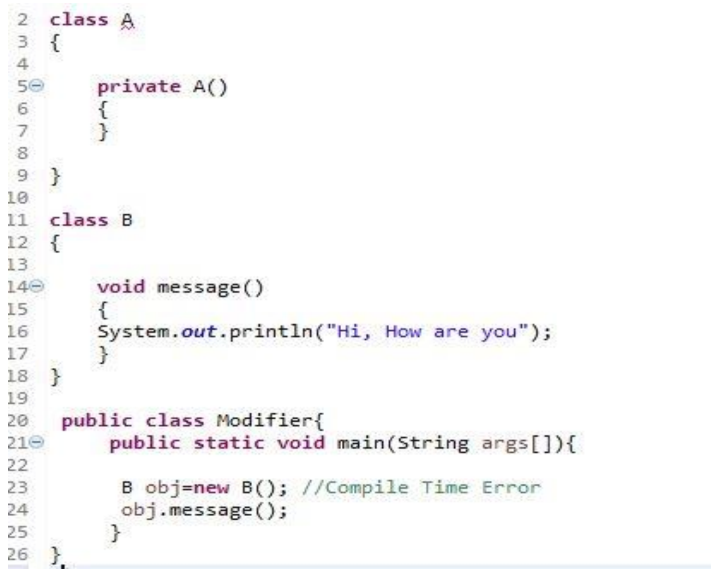
Default Access Modifiers:

The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

b).

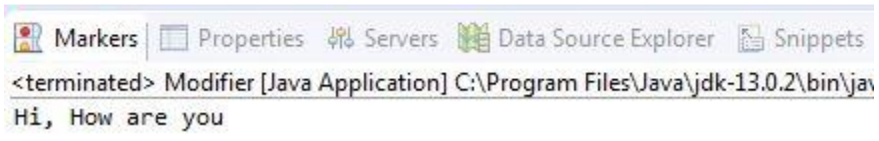
Program for Private Access modifiers:

Code:



```
class A
{

    private A()
    {
    }

}

class B
{

    void message()
    {
    System.out.println("Hi, How are you");
    }
}

 public class Modifier{
    public static void main(String args[]){

     B obj=new B(); //Compile Time Error
     obj.message();
    }
}
```

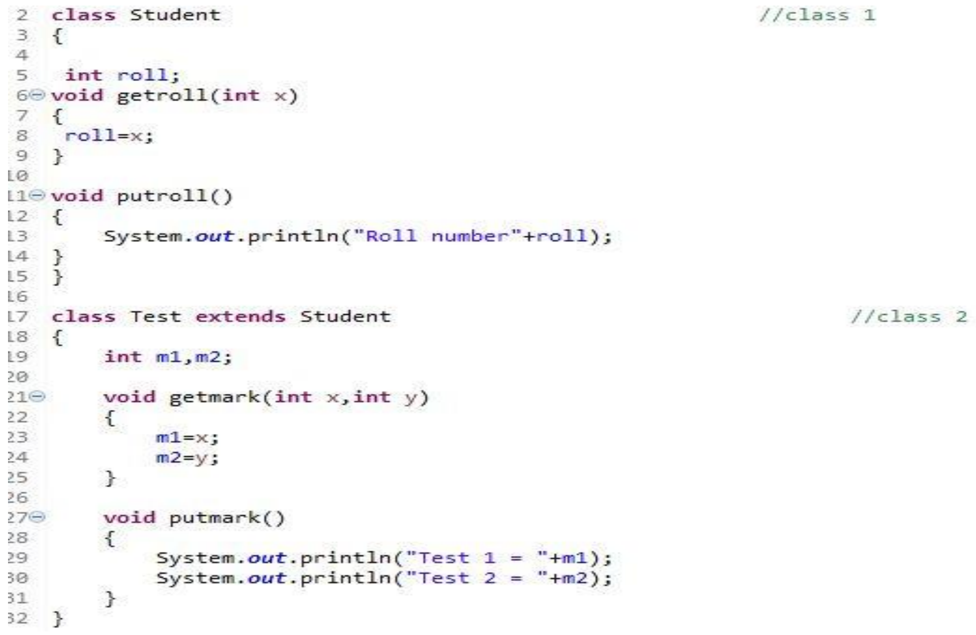
Out put:



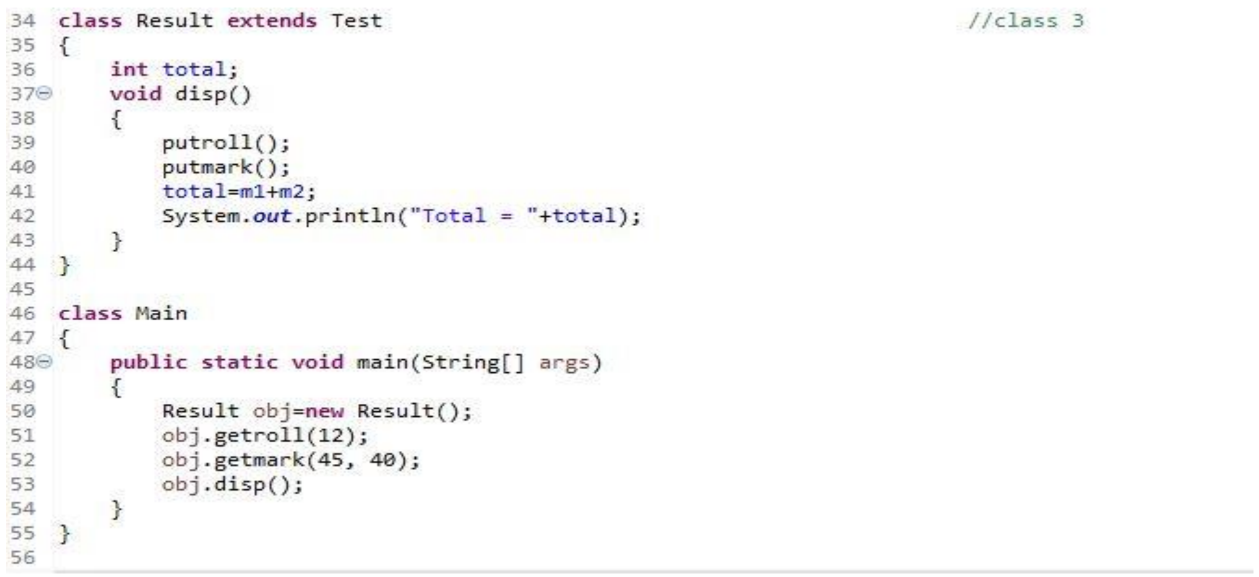
```
<terminated> Modifier [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\jav
Hi, How are you
```

Program for Default Access modifiers:

Code:

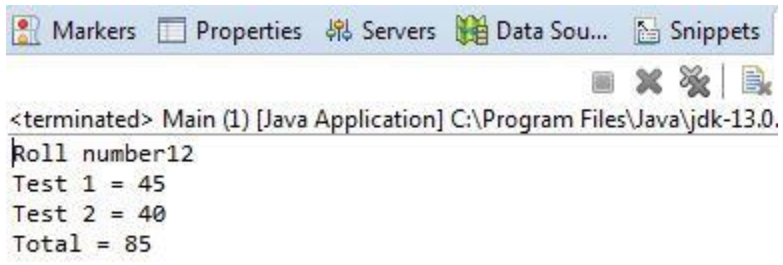


```
class Student                                        //class 1
{

 int roll;
void getroll(int x)
{
 roll=x;
}

void putroll()
{
    System.out.println("Roll number"+roll);
}
}

class Test extends Student                                   //class 2
{
    int m1,m2;

    void getmark(int x,int y)
    {
        m1=x;
        m2=y;
    }

    void putmark()
    {
        System.out.println("Test 1 = "+m1);
        System.out.println("Test 2 = "+m2);
    }
}
```



```
class Result extends Test                                         //class 3
{
    int total;
    void disp()
    {
        putroll();
        putmark();
        total=m1+m2;
        System.out.println("Total = "+total);
    }
}

class Main
{
    public static void main(String[] args)
    {
        Result obj=new Result();
        obj.getroll(12);
        obj.getmark(45, 40);
        obj.disp();
    }
}
```

Output:



```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-13.0.
Roll number12
Test 1 = 45
Test 2 = 40
Total = 85
```

Question 2:

a)

Public access modifiers:

The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package. Public access modifier access everywhere as you want.

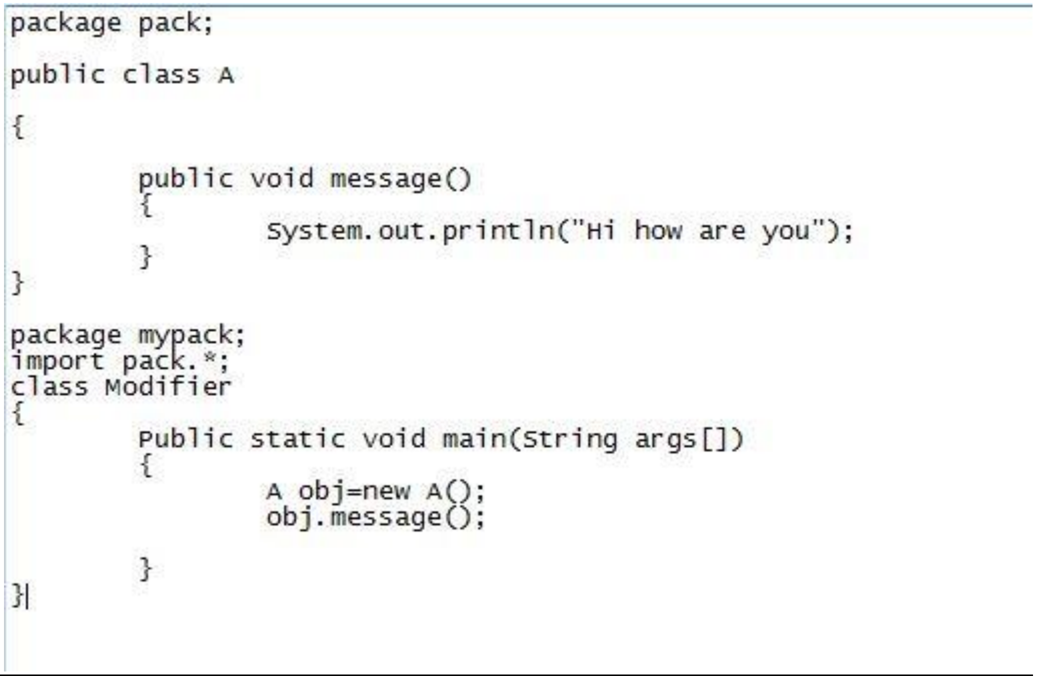
Protected access modifiers:

Variables, methods, and constructors, which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class. The protected access modifier cannot be applied to class and interfaces.

b).

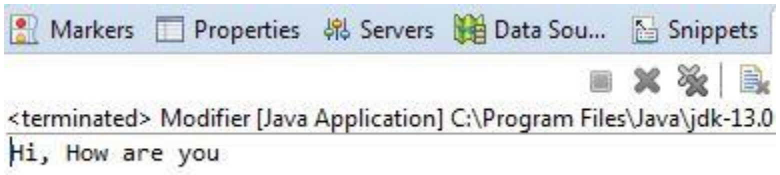
Program for public:

Code:



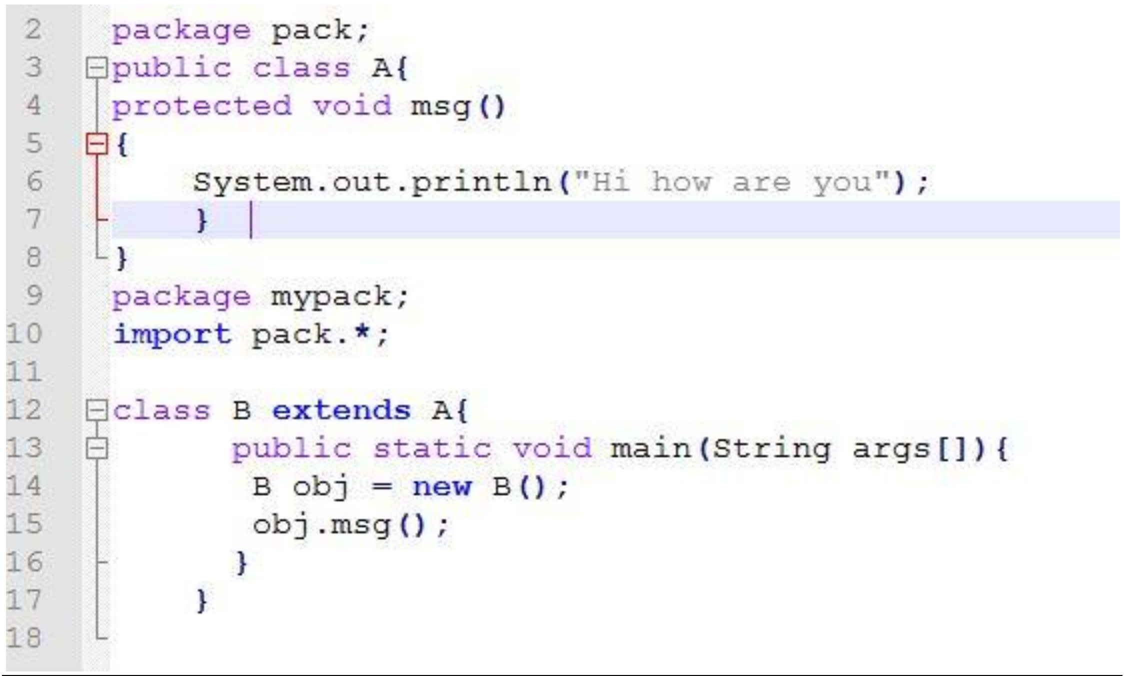
```
package pack;

public class A

{

        public void message()
        {
                System.out.println("Hi how are you");
        }
}

package mypack;
import pack.*;
class Modifier
{
        Public static void main(String args[])
        {
                A obj=new A();
                obj.message();

        }
}
```

Output:



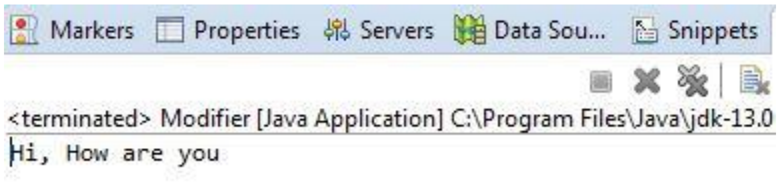
```
<terminated> Modifier [Java Application] C:\Program Files\Java\jdk-13.0
Hi, How are you
```

Program for protected class:



```
package pack;
public class A{
protected void msg()
{
    System.out.println("Hi how are you");
    }
}
package mypack;
import pack.*;

class B extends A{
      public static void main(String args[]){
       B obj = new B();
       obj.msg();
      }
    }

```

Output:



```
<terminated> Modifier [Java Application] C:\Program Files\Java\jdk-13.0
Hi, How are you
```

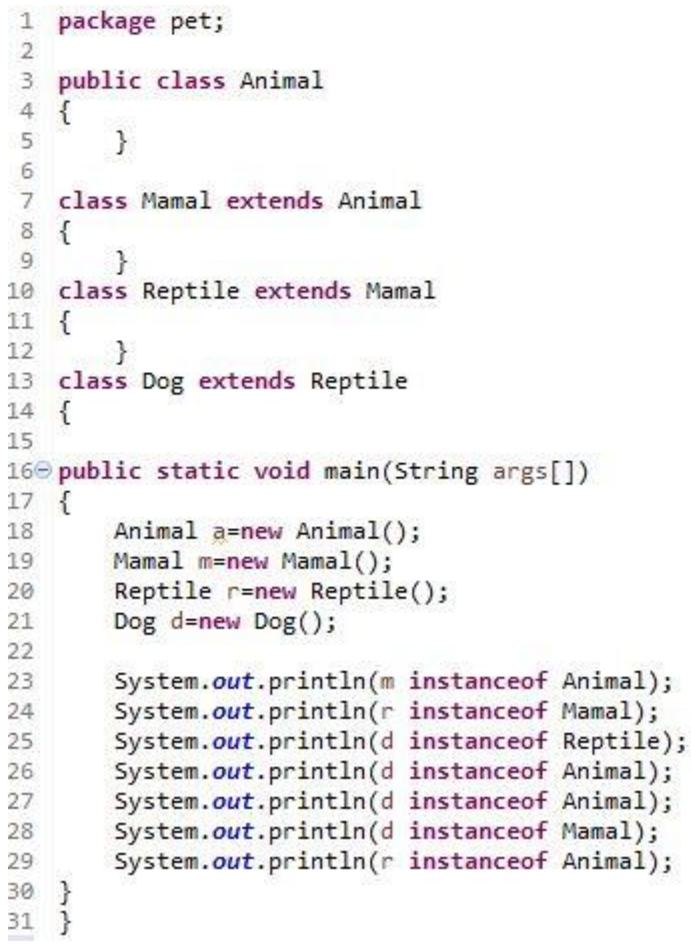
Question 3:

a)

Inheritance:

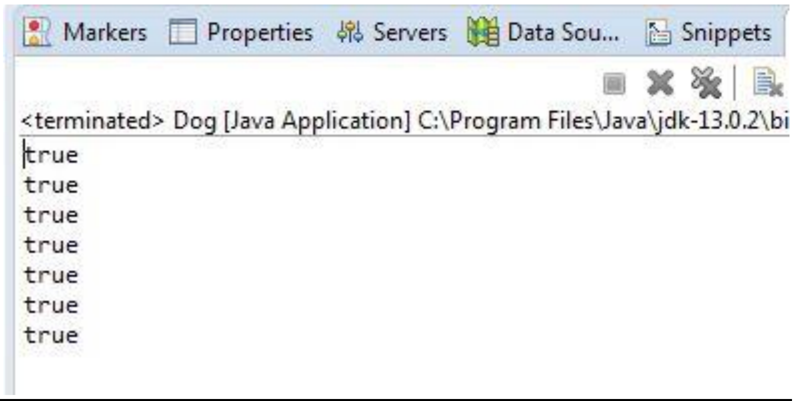
Inheritance in Java Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features (fields and methods) of another class. ... The subclass can add its own fields and methods in addition to the super class fields and methods.

b)



```java
package pet;

public class Animal
{
    }

class Mamal extends Animal
{
    }
class Reptile extends Mamal
{
    }
class Dog extends Reptile
{

public static void main(String args[])
{
    Animal a=new Animal();
    Mamal m=new Mamal();
    Reptile r=new Reptile();
    Dog d=new Dog();

    System.out.println(m instanceof Animal);
    System.out.println(r instanceof Mamal);
    System.out.println(d instanceof Reptile);
    System.out.println(d instanceof Animal);
    System.out.println(d instanceof Animal);
    System.out.println(d instanceof Mamal);
    System.out.println(r instanceof Animal);
}
}
```

Output:



```
<terminated> Dog [Java Application] C:\Program Files\Java\jdk-13.0.2\bi
true
true
true
true
true
true
true
```

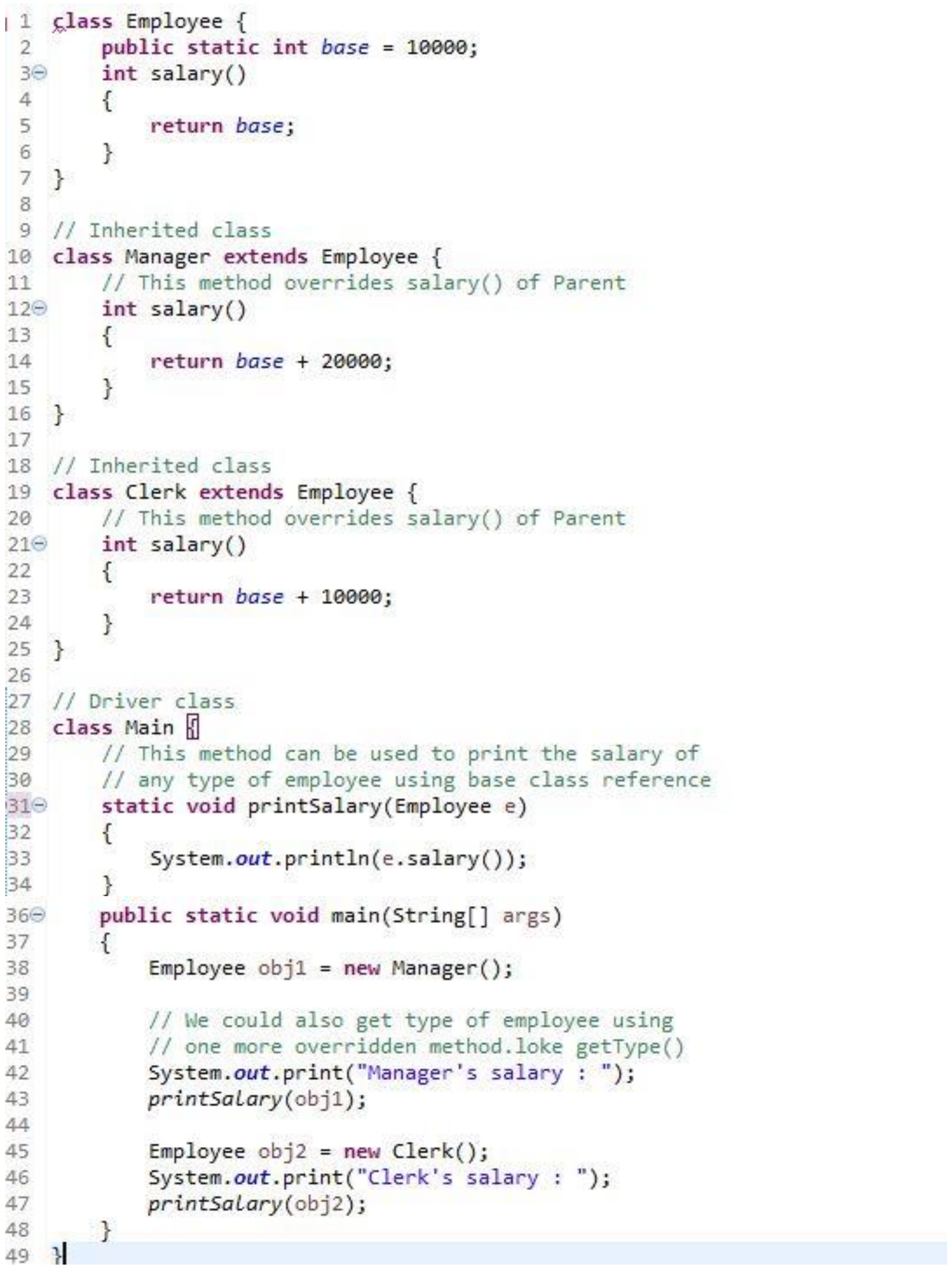
Question 4:

a)

Polymorphism:

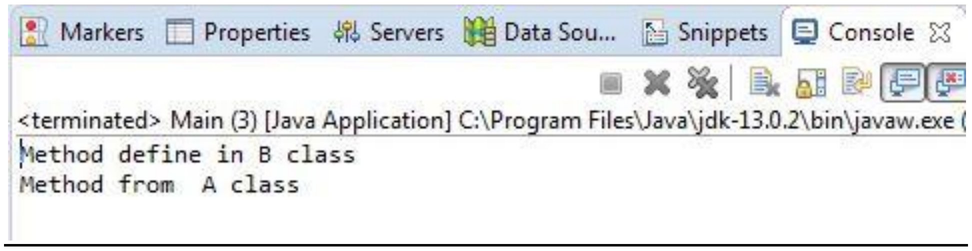
Polymorphism in Java is a concept by which we can perform a single action in different ways. We can perform polymorphism in java by method overloading and method overriding. If you overload a static method in Java, it is the example of compile time polymorphism. Here, we will focus on runtime polymorphism in java.

b.



```
class Employee {
    public static int base = 10000;
    int salary()
    {
        return base;
    }
}

// Inherited class
class Manager extends Employee {
    // This method overrides salary() of Parent
    int salary()
    {
        return base + 20000;
    }
}

// Inherited class
class Clerk extends Employee {
    // This method overrides salary() of Parent
    int salary()
    {
        return base + 10000;
    }
}

// Driver class
class Main {
    // This method can be used to print the salary of
    // any type of employee using base class reference
    static void printSalary(Employee e)
    {
        System.out.println(e.salary());
    }
    public static void main(String[] args)
    {
        Employee obj1 = new Manager();

        // We could also get type of employee using
        // one more overridden method.loke getType()
        System.out.print("Manager's salary : ");
        printSalary(obj1);

        Employee obj2 = new Clerk();
        System.out.print("Clerk's salary : ");
        printSalary(obj2);
    }
}
```

Output:



```
<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (
Method define in B class
Method from  A class
```

Question 5:

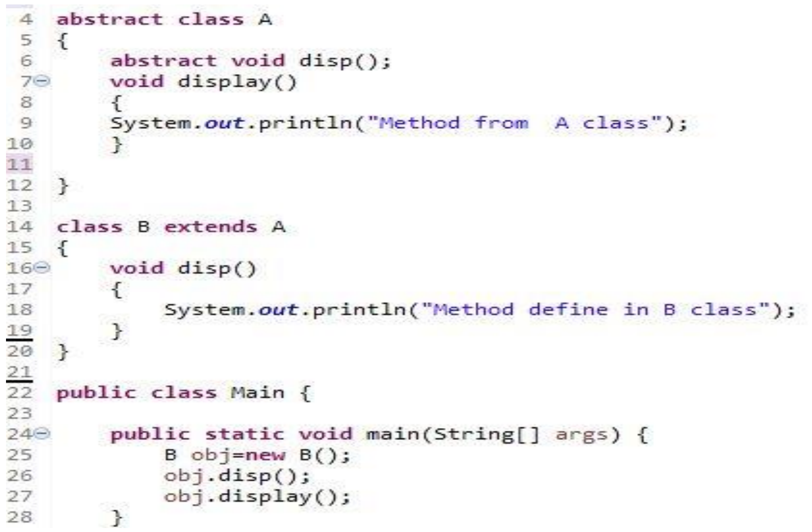
a)

Abstraction:

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message.

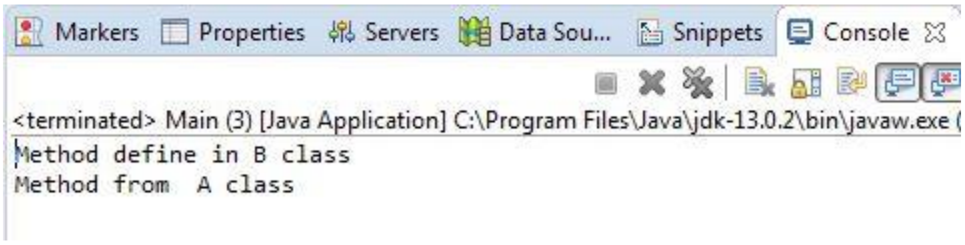
b)

code:



```java
abstract class A
{
    abstract void disp();
    void display()
    {
    System.out.println("Method from  A class");
    }

}

class B extends A
{
    void disp()
    {
        System.out.println("Method define in B class");
    }
}

public class Main {

    public static void main(String[] args) {
        B obj=new B();
        obj.disp();
        obj.display();
    }
```

Output:



```
<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (
Method define in B class
Method from  A class
```