PROGRAMMING IN JAVA

Paper Code – 32341201

BSc (Hons) Computer Science CBCS

Semester – II

Anshul Verma

Roll No – 19/78065

Practical Assignment

Q1. What is the difference between function overloading and constructor overloading in Java?

Write a suitable program to illustrate the following:

a) Default constructor
b) Parameterized constructor
c) Method overloading with different number of parameters
d) Method overloading with type of parameters
e) Method overloading with difference sequence of parameters

Ans:

Difference :

Method Overloading :	Constructor Overloading :
1) If a class has multiple methods having same name and different parameters then it is known as Method Overloading. 2) Method overloading can't be performed by changing the return type of the method only. 3) The access can be public, private or protected. 4) Method overloading is used to increase the readability and flexibility of the program.	1) Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. 2) Constructor does not have a return type. 3) The access should always be public. 4) Constructor overloading is used to provide multiple ways to a user to initialise class objects.

Program Code :

```
class User {
    String name, address;
    boolean isRegistered;
    User() {
        this.name = "Unknown";
        this.isRegistered = false;
        System.out.println("User not registered! //default constructor");
    }
    User(String name) {
        this.name = name;
        this.isRegistered = true;
        System.out.println("User " + name + " registered! //Parameterized c
onstructor");
    }
    void setAddress(String details) {
        this.address = details;
    }
    void setAddress(String city, String zip) {
        this.address = city + " " + (zip);
    }
    void setAddress(String city, int zip) {
        this.address = city + " " + Integer.toString(zip);
    }
    void setAddress(int zip, String city) {
        this.address = city + " " + Integer.toString(zip);
    }
}
public class Test {
    public static void main(String[] args) {
        System.out.println();
        User a = new User();
        User b = new User("Abhishek");
```

```
        System.out.println("\nMethod Overloading with no of parameters : ")
;

        b.setAddress("Delhi 110011");

        System.out.println("Address : " + b.address + " //1 parameter of ty
pe string");

        b.setAddress("Delhi", "110011");

        System.out.println("Address : " + b.address + " //2 parameters of t
ype string_string");

        System.out.println("\nMethod Overloading with type of parameters :
");

        b.setAddress("Delhi", "110011");

        System.out.println("Address : " + b.address + " //2 parameters of t
ype string_string");

        b.setAddress("Delhi", 110011);

        System.out.println("Address : " + b.address + " //2 parameters of t
ype string_int");

        System.out.println("\nMethod Overloading with different sequence of
 parameters : ");

        b.setAddress(110011, "110011");

        System.out.println("Address : " + b.address + " //2 parameters of t
ype int_string");

        b.setAddress("Delhi", 110011);

        System.out.println("Address : " + b.address + " //2 parameters of t
ype string_int");

    }

}
```

Output :

```
User not registered! //default constructor
User Abhishek registered! //Parameterized constructor

Method Overloading with no of parameters :
Address : Delhi 110011 //1 parameter of type string
Address : Delhi 110011 //2 parameters of type string_string

Method Overloading with type of parameters :
Address : Delhi 110011 //2 parameters of type string_string
Address : Delhi 110011 //2 parameters of type string_int

Method Overloading with different sequence of parameters :
Address : 110011 110011 //2 parameters of type int_string
Address : Delhi 110011 //2 parameters of type string_int
```

Q2. Write the advantages of using packages in Java. Write a suitable program that illustrates different levels of protection in classes/subclasses to same package or different packages.

Ans:

Advantages of using packages in java :

1) Prevent name Conflicts : Using packages prevents naming conflicts e.g. there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee. It helps us to uniquely identify a class.
2) Maintenance : Using packages makes searching/locating and usage of classes, interfaces, enumerations easier.
3) Controlled Access: Packages are used for providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
4) Data Encapsulation : Packages are a way to hide classes, preventing other programs from accessing classes that are meant for internal use only.
5) Re-usability : The class contained in the packages of another program can be easily reused.

Program illustrating levels of protection :

p1/Protection.java :

```
package p1;


public class Protection {
    int n = 1;
    private int n_pri = 2;
    protected int n_pro = 3;
    public int n_pub = 4;

    public Protection() {
        System.out.println("base constructor");
        System.out.println("n = " + n);
        System.out.println("n_pri = " + n_pri + "//private variable");
        System.out.println("n_pro = " + n_pro + "//protected variable");
        System.out.println("n_pub = " + n_pub + "//public variable");
    }
```

```
}
```

p1/Derived.java :

```
package p1;

class Derived extends Protection {
    Derived() {
        System.out.println("derived constructor");
        System.out.println("n = " + n);
        // class only
        // System.out.println("n_pri = "4 + n_pri + "//private variable");
        System.out.println("n_pro = " + n_pro + "//protected variable");
        System.out.println("n_pub = " + n_pub + "//public variable");
    }
}
```

p1/SamePackage.java :

```
package p1;

class SamePackage {
    SamePackage() {
        Protection p = new Protection();
        System.out.println("same package constructor");
        System.out.println("n = " + p.n);
        // class only
        // System.out.println("n_pri = " + p.n_pri + "//private variable");
        System.out.println("n_pro = " + p.n_pro + "//protected variable");
        System.out.println("n_pub = " + p.n_pub + "//public variable");
    }
}
```

p2/Protection2.java :

```
package p2;

class Protection2 extends p1.Protection {
```

```
    Protection2() {
        System.out.println("derived other package constructor");
        // class or package only
        // System.out.println("n = " + n + "//default variable");
        // class only
        // System.out.println("n_pri = " + n_pri + "//private variable");
        System.out.println("n_pro = " + n_pro + "//priotected variable");
        System.out.println("n_pub = " + n_pub + "//public variable");
    }
}
```

p2/OtherPackage.java :

```
package p2;

class OtherPackage {
    OtherPackage() {
        p1.Protection p = new p1.Protection();
        System.out.println("other package constructor");
        // class or package only
        // System.out.println("n = " + p.n + "//default variable");
        // class only
        // System.out.println("n_pri = " + p.n_pri + "//private variable");
        // class, subclass or package only
        // System.out.println("n_pro = " +p.n_pro+ "//protected variable");
        System.out.println("n_pub = " + p.n_pub + "//public variable");
    }
}
```

Test file for package **p1 :**

```
// Demo package p1.
package p1;

// Instantiate the various classes in p1.
public class Demo {
```

```
    public static void main(String args[]) {
        Protection ob1 = new Protection();
        Derived ob2 = new Derived();
        SamePackage ob3 = new SamePackage();
    }
}
```

Output :

```
base constructor
n = 1//default variable
n_pri = 2//private variable
n_pro = 3//protected variable
n_pub = 4//public variable
base constructor
n = 1//default variable
n_pri = 2//private variable
n_pro = 3//protected variable
n_pub = 4//public variable
derived constructor
n = 1
n_pro = 3//protected variable
n_pub = 4//public variable
base constructor
n = 1//default variable
n_pri = 2//private variable
n_pro = 3//protected variable
n_pub = 4//public variable
same package constructor
n = 1//default variable
n_pro = 3//protected variable
n_pub = 4//public variable
```

Test file for package **p2 :**

```
// Demo package p2.
package p2;
// Instantiate the various classes in p2.
public class Demo {
      public static void main(String args[]) {
            Protection2 ob1 = new Protection2();
            OtherPackage ob2 = new OtherPackage();
      }
}
```

Output :

```
base constructor
n = 1//default variable
n_pri = 2//private variable
n_pro = 3//protected variable
n_pub = 4//public variable
derived other package constructor
n_pro = 3//priotected variable
n_pub = 4//public variable
base constructor
n = 1//default variable
n_pri = 2//private variable
n_pro = 3//protected variable
n_pub = 4//public variable
other package constructor
n_pub = 4//public variable
```

=> We can see the following from the both outputs above :

Default, private, protected and public declared variables are printed from the base class if accessed in the same package.

Only the default, protected and public declared variables are printed from the base class if accessed in derived class in the same package.

Default, private, protected and public declared variables are printed from the base class if accessed in the other package.

Only the protected and public declared variables are printed from the base class if accessed in class extended in other (different) package.

Only the public declared variable is printed if accessed in derived class of a class extended from the base class in other (different) package.