CS261 ASSIGNMENT 3

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202052307

SECTION:

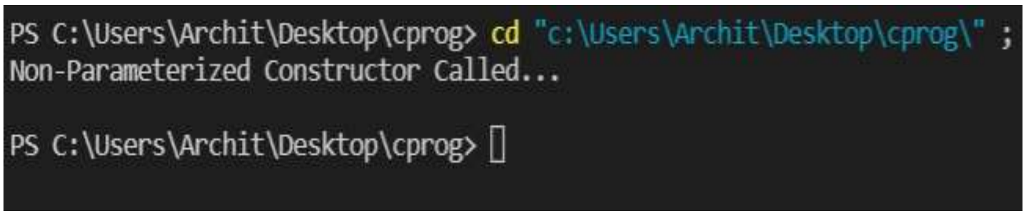
A

1. **Explain types of constructors with JAVA program. (Write small code that gives definition)**

   There are 4 different types of constructors -:

   - **Non-Parameterized Constructor** -: When a constructor doesn't have any parameter, it is said to be Non-Parameterized.

```
public class Area{

    double area; //instance variable of class Area

    //Non-Parameterized Constructor - with no parameters
    public Area(){
        System.out.println("Non-Parameterized Constructor 
Called...");
    }

    public static void main(String[] args){
        //creating object using Non-Parameterized Constructor
        Area obj = new Area();
        System.out.println();
    }

}
```

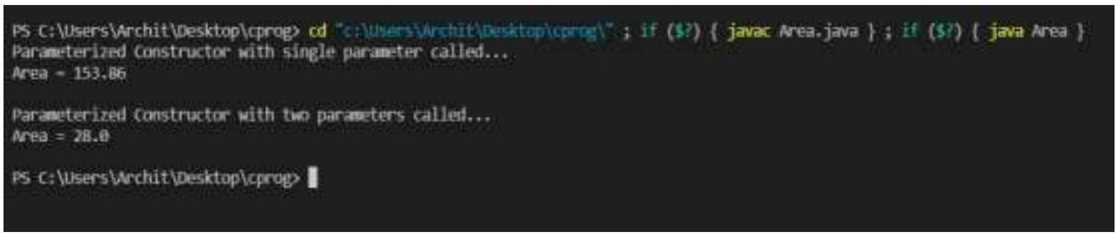


```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ;
Non-Parameterized Constructor Called...

PS C:\Users\Archit\Desktop\cprog> []
```

   - **Parameterized Constructor** -: A constructor with a specific number of parameters (more than or equal to 1) is known as parameterized constructor. The parameterized constructor is used to provide values to attributes of distinct objects.

```
public class Area{

    double area; //instance variable of class Area

    //Parameterized Constructor -: 1 parameter
    public Area(double r){ //with a single parameter
        System.out.println("Parameterized Constructor with single 
parameter called...");
```

```
        this.area = 3.14 * r * r;
    }

    //Parameterized Constructor -: 2 parameters
    public Area(double l, double b){ //constructor overloading
        System.out.println("Parameterized Constructor with two
parameters called...");
        this.area = l * b;
    }

    public static void main(String[] args){

        //creating an object using Parameterised constructor
        Area circle = new Area(7);
        System.out.println("Area = " +circle.area);
        System.out.println();

        //creating an object using Patameterized Constructor with
two parameters
        Area rectangle = new Area(7, 4);
        System.out.println("Area = "+rectangle.area);
        System.out.println();
    }
}
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Area.java } ; if ($?) { java Area }
Parameterized Constructor with single parameter called...
Area = 153.86

Parameterized Constructor with two parameters called...
Area = 28.0

PS C:\Users\Archit\Desktop\cprog> 
```

- **Default Constructor** -: When no constructor is defined explicitly by the programmer, the JAVA compiler creates a non-parameterized constructor itself. The instance variable of the object created are set to default in such a case i.e 0, null, etc.

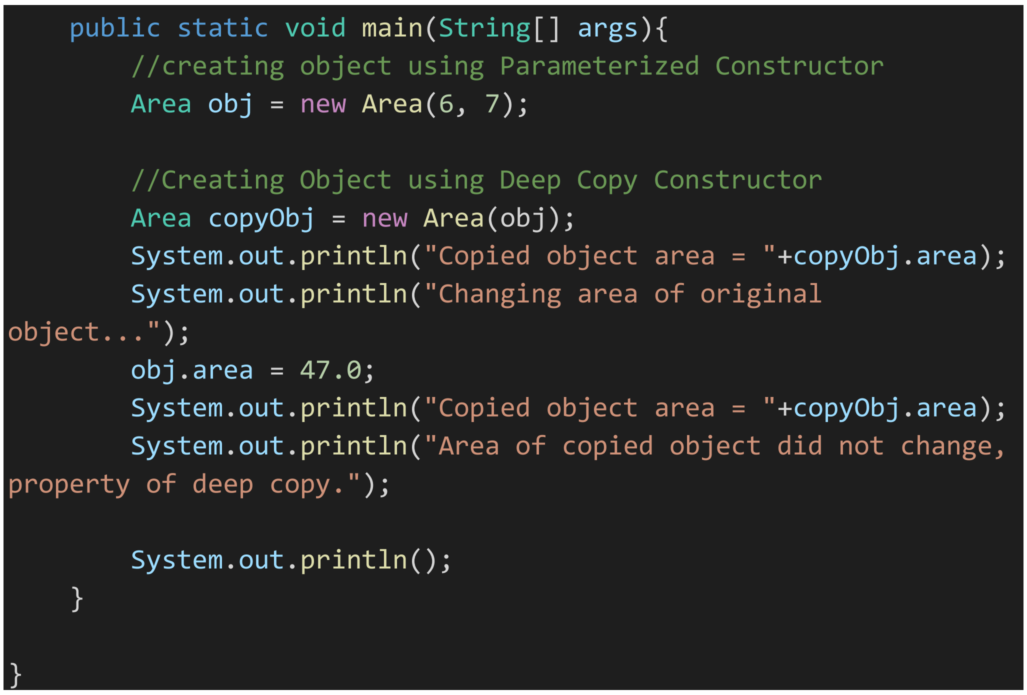
```
public class Area{

    double area; //instance variable of class Area

    //No constructor written explicitly
    //compiler will create a default constructor itself

    public static void main(String[] args){
        //creating object using Default Constructor
        Area obj = new Area();

        System.out.println("Object created using default 
constructor...");
        System.out.println();
    }
}
```

- **Copy Constructor** -: When a constructor creates an object using another constructor, it is known as copy constructor. There are two types of copy constructors -:

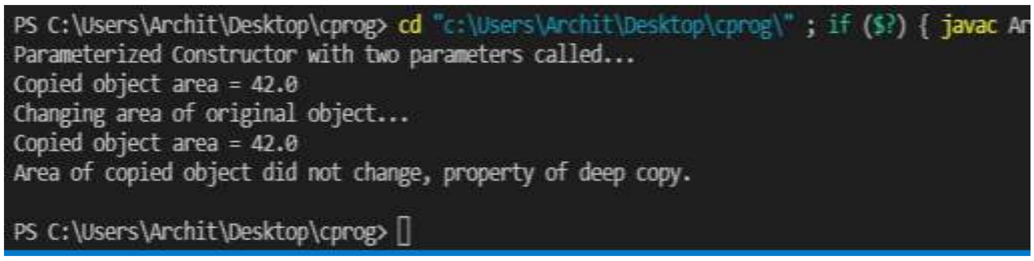
  Shallow Copy -: It creates a copy of an object by copying data of all member variables. Both the new created object and already existing object reference to the same location and hence changes made in one affects the other.

  Deep Copy -: It is also known as user-defined copy constructor. It dynamically allocates memory for the new object and copies entities of the existing object to this new object. Since, both objects have different memory locations, changes in one doesn't affect the other.

```
import java.util.*;

public class Area{

    double area; //instance variable of class Area

    //Parameterized Constructor -: 2 parameters
    public Area(double l, double b){ //constructor overloading
        System.out.println("Parameterized Constructor with two 
parameters called...");
        this.area = l * b;
    }

    //Deep Copy Constructor
    public Area(Area obj){
        this.area = obj.area;
    }
```



```
    public static void main(String[] args){
        //creating object using Parameterized Constructor
        Area obj = new Area(6, 7);

        //Creating Object using Deep Copy Constructor
        Area copyObj = new Area(obj);
        System.out.println("Copied object area = "+copyObj.area);
        System.out.println("Changing area of original
object...");
        obj.area = 47.0;
        System.out.println("Copied object area = "+copyObj.area);
        System.out.println("Area of copied object did not change,
property of deep copy.");

        System.out.println();
    }

}
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Ar
Parameterized Constructor with two parameters called...
Copied object area = 42.0
Changing area of original object...
Copied object area = 42.0
Area of copied object did not change, property of deep copy.

PS C:\Users\Archit\Desktop\cprog> 
```

Apart from these four constructors, there is another concept known as constructor overloading. A class can have as many constructors as we want, with the name similar to that of class name, and the parameters must be different for each constructor. This is known as constructor overloading (as shown in Parameterized Constructor example).

2. **Explain types of method overloading and write a program in JAVA for any one of them.**

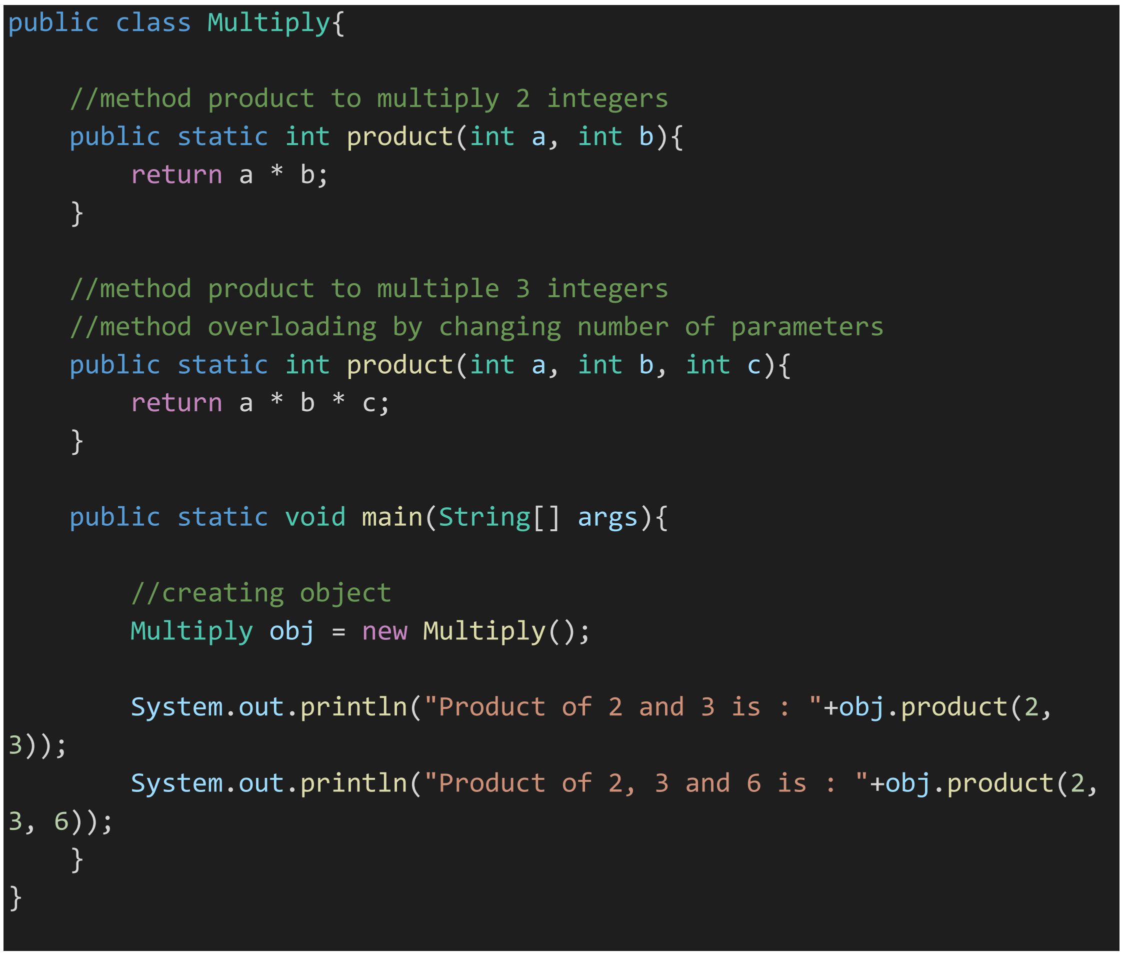
   Method overloading can be done by changing:

   - **<u>The number of parameters</u>**
     for example,
     ⇨ int multiply(int a, int b)
     ⇨ int multiply(int a, int b, int c)
     In the former one, there are only two parameters, while in the latter one there are three parameters.
   - **<u>The data types of parameters</u>**
     for example,
     ⇨ double percentage(int marksObtained, int maximumMarks)
     ⇨ double percentage(float marksObtained, int maximumMarks)
     In the former one, the variable marksObtained is 'int' type while in the latter one it is 'float' type.
   - **<u>The ordering of parameters</u>**
     for example,
     ⇨ void Report(String name, String bloodGroup, int plateletCount)
     ⇨ void Report(String bloodGroup, int plateletCount, String name)
     Although the parameters are same in both the methods, but the ordering of the parameters is different.

   <u>Note</u> -: If the method signature is same and only return type is different, method overloading will not occur and the compiler will throw an error as the return value alone is not sufficient for the compiler to figure out which function it has to call.

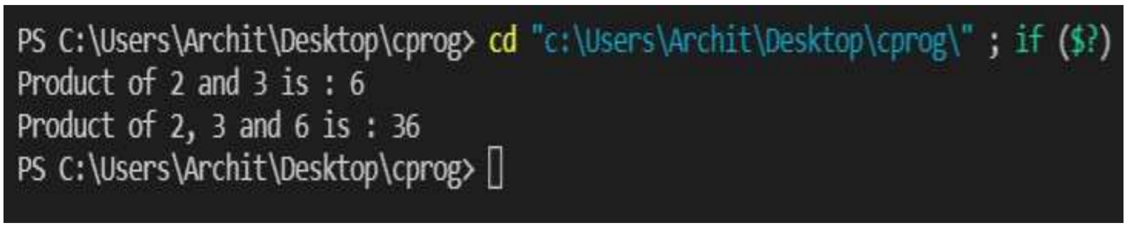
<u>CODE</u>

Constructing a class Multiply, with a method, product(). Method Overloading is done by changing the number of parameters(two integers in one method while three in other).

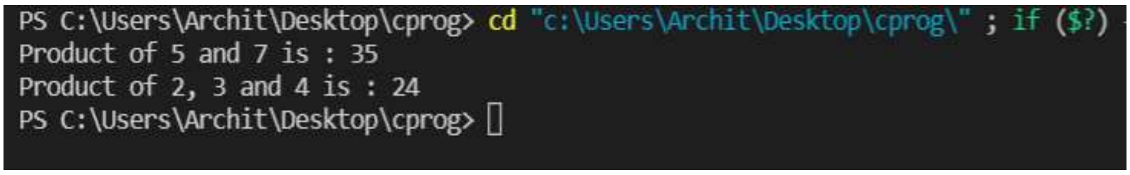


```java
public class Multiply{

    //method product to multiply 2 integers
    public static int product(int a, int b){
        return a * b;
    }

    //method product to multiple 3 integers
    //method overloading by changing number of parameters
    public static int product(int a, int b, int c){
        return a * b * c;
    }

    public static void main(String[] args){

        //creating object
        Multiply obj = new Multiply();

        System.out.println("Product of 2 and 3 is : "+obj.product(2,
3));
        System.out.println("Product of 2, 3 and 6 is : "+obj.product(2,
3, 6));
    }
}
```

<u>Output</u>



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?)
Product of 2 and 3 is : 6
Product of 2, 3 and 6 is : 36
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?)
Product of 5 and 7 is : 35
Product of 2, 3 and 4 is : 24
PS C:\Users\Archit\Desktop\cprog> 
```

3. **State whether the given below are valid or invalid cases of method overloading. Give a short explanation.**

   Case 1:
   int demo(int a, int b, float c)
   int demo(int var1, int var2, float var3)
   Case 2:
   int demo(int a, int b)
   int demo(float var1, float var2)
   Case 3:
   int demo(int a, int b)
   int demo(int num)
   Case 4:
   float demo(int a, float b)
   float demo(float var1, int var2)
   Case 5:
   int demo(int a, int b)
   float demo(int var1, int var2)

   <u>Explanation and Answers</u>

   Case 1: Since the number of parameters, their data types and their order is same, it is an invalid case of method overloading.

   Case 2: Since the data types of the parameters are different in both the methods, it is a valid case of method overloading.

   Case 3: Since the number of the parameters are different in both the methods(2 in upper one and 1 in lower one), it is a valid case of method overloading.

   Case 4: Since the order of the parameters are different in both the methods(int then float in upper one, and reverse in lower one), it is a valid case of method overloading.

Case 5: Since the number of parameters, their data types and their order is same, it is an invalid case of method overloading. Although the return type is different, but the compiler will not be able to figure out which function it has to call.

4. Correct the code for overloading methods

Code:

```
1.      class Demo1{
2.
3.          public double myMethod(int num1, int num2){
4.              System.out.println("First myMethod of class Demo");
5.              return num1+num2;
6.          }
7.
8.          public int myMethod(int var1, int var2){
9.              System.out.println("Second myMethod of class Demo");
10.             return var1-var2;
11.         }
12.
13.     }
14.
15.     class Demo2{
16.         public static void main(String args[]){
17.
18.             Demo2 obj2= new Demo2();
19.             obj2.myMethod(10,10);
20.             obj2.myMethod(20,12);
21.         }
22.     }
```

Following changes have been made and the corrected code is provided below:

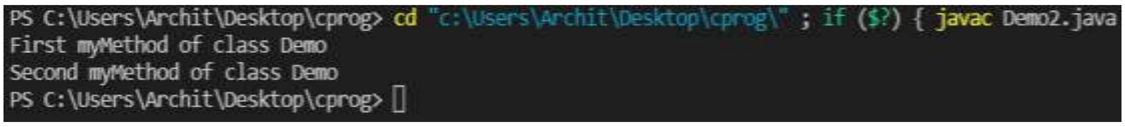
- Since both the methods 'myMethod' have same number of parameters, same order and same data types of parameters, therefore, it is an invalid case of method overloading. To correct it, the data type of parameters passed in the 'myMethod' written above (Line 3) has been changed to 'double' from 'int'.
- Class Demo2 must be 'public' as it contains the 'main' method.(Line 15).
- Object of class Demo1 should've been created in line 18. It's name(i.e obj2) doesn't have any effect so it is left unchanged.

- Not a required change, done so that both the methods runs at least once in Line 19. (10, 10) were passed earlier, now (10.0, 10.0) are passed.

Corrected Code

```
1.      class Demo1{
2.
3.          public double myMethod(double num1, double num2){
4.              System.out.println("First myMethod of class Demo");
5.              return num1+num2;
6.          }
7.
8.          public int myMethod(int var1, int var2){
9.              System.out.println("Second myMethod of class Demo");
10.             return var1-var2;
11.         }
12.
13.     }
14.
15.     public class Demo2{
16.         public static void main(String args[]){
17.
18.             Demo1 obj2= new Demo1();
19.             obj2.myMethod(10.0,10.0);
20.             obj2.myMethod(20,12);
21.         }
22.     }
```

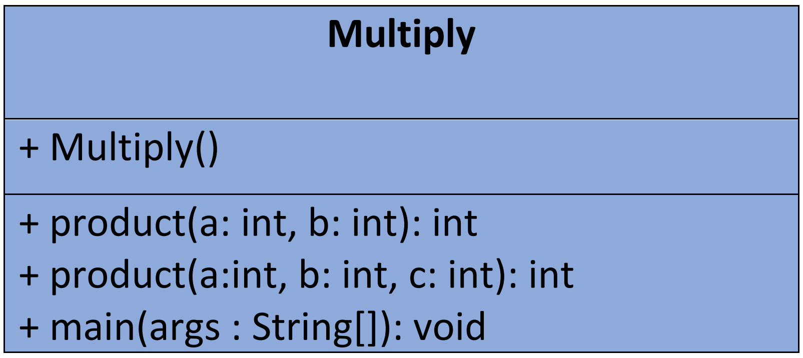
Output



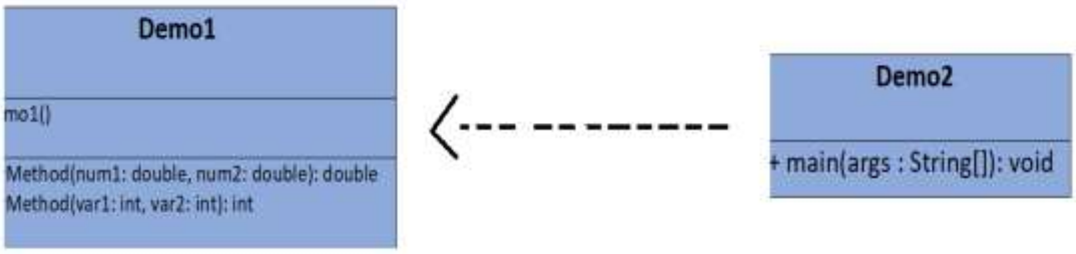
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Demo2.java
First myMethod of class Demo
Second myMethod of class Demo
PS C:\Users\Archit\Desktop\cprog> 
```

5. **Prepare a class diagram for Q2 and Q4 of this assignment.**

Q.2


Multiply
+ Multiply()
+ product(a: int, b: int): int
+ product(a:int, b: int, c: int): int
+ main(args : String[]): void

Q.4


Demo1
mo1()
Method(num1: double, num2: double): double
Method(var1: int, var2: int): int
Demo2
+ main(args : String[]): void