



## ASSIGNMENT 2

Module 2: Object Oriented Programming in JAVA

### ABSTRACT

In this assignment, I have completed 42 questions on java basic topics like array, star pattern, operators, classes and object, access specifier, SIB, Inheritance, Polymorphism, Encapsulation, this keyword, super keyword, command line arguments, abstract class, interface, non-sub class, call by reference – call by value, Exception handling (and its five keyword namely try – catch – finally – throw – throws), boxing and unboxing, Wrapper class, == and equals, and last but not the least some mind busting problems. Every question starts from new page along with its code and output.

sarang deodhar

220350320026

---

**Q1. Write a Java program for Sorting elements of an array.**

Code:

```
package assignment2;

import java.util.Scanner;

public class Sorting {
    static int n;
    static int j;
    static int i;
    static int temp;
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.println("Enter the number of elements : ");
        n = sc.nextInt();           //taking number of element as n
        int array[] = new int[n];
        System.out.println("Enter the elements : ");
        for(i = 0; i < n ; i++) {
            System.out.print("> ");
            array[i] = sc.nextInt(); //taking all n elements
        }
        for(i = 0 ; i < n ; i++) {
            //checking each element i with every next element j
            for(int j = i+1 ; j < n ; j++) {
                if(array[i] > array[j]) {
                    //swapping if next number is bigger
                    temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
            }
        }
        System.out.println("\nSorted numbers : ");
        System.out.println("-----");
        //printing sorted array
        for(int i : array)
            System.out.println(i);
        sc.close();
    }
}
```

**Output:**

```
1 package assignment2;
2
3 import java.util.Scanner;
4
5 public class Sorting {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Enter the number of elements : ");
9         int n = sc.nextInt();
10        System.out.println("Enter the elements : ");
11        int arr[] = new int[n];
12        for (int i = 0; i < n; i++) {
13            arr[i] = sc.nextInt();
14        }
15        System.out.println("Sorted numbers : ");
16        for (int i = 0; i < n; i++) {
17            System.out.print(arr[i] + " ");
18        }
19    }
20 }
```

Enter the number of elements :  
5  
Enter the elements :  
=>23  
=>45  
=>12  
=>65  
=>95  
  
Sorted numbers :  
-----  
12 23 45 65 95

---

**Q2. Write a Java Program for Star Patterns.**

a. **Square Star Pattern**

```
*****
*****
*****
*****
```

b. **Triangle Star Pattern**

```
*
```

```
**
```

```
***
```

```
****
```

c. **Inverted Pyramid**

```
*****
```

```
****
```

```
***
```

```
*
```

**Code:**

**Part – A**

```
package assignment2;

import java.util.Scanner;

public class StarPattern1 {
    @SuppressWarnings("unused")
    private static int i = 0;
    @SuppressWarnings("unused")
    private static int j = 0;
    private static int n = 0;
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of row : ");
        n = sc.nextInt();
        for(int i = 0 ; i < n ; i++) {
            for(int j = 0 ; j < n ; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

---

Part – B

```
package assignment2;

import java.util.Scanner;

public class StarPattern2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows");
        int n = sc.nextInt();
        for(int i = 0 ; i < n ; i++) {
            for(int j = 0 ; j <= i ; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

Part – C

```
package assignment2;

import java.util.Scanner;

public class StarPattern3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows : ");
        int n = sc.nextInt();
        for(int i = 0 ; i <= n ; i++) {
            for(int j = 0 ; j < i ; j++) {
                System.out.printf(" ");
            }
            for(int k = i ; k < (2*n-(i+1)) ; k++) {
                System.out.print("*");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

**Output:**

**Part – A**

The screenshot shows the Eclipse IDE interface with the project 'assignment2' open. The 'StarPattern1.java' file is selected in the Project Explorer. The code defines a static method 'main' that prints a 5x5 star pattern. The output window shows the printed pattern.

```
1 package assignment2;
2
3 import java.util.Scanner;
4
5 public class StarPattern1 {
6     @SuppressWarnings("unused")
7     private static int i = 0;
8     @SuppressWarnings("unused")
9     private static int j = 0;
10    private static int n = 0;
11    public static void main(String[] args) {
12        Enter number of row : 5
13        *****
14        *****
15        *****
16        *****
17        *****
18    }
19}
```

**Part – B**

The screenshot shows the Eclipse IDE interface with the project 'assignment2' open. The 'StarPattern2.java' file is selected in the Project Explorer. The code defines a static method 'main' that prints a star pattern based on user input. The output window shows the printed pattern.

```
1 package assignment2;
2
3 import java.util.Scanner;
4
5 public class StarPattern2 {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Enter the number of rows");
9         int n = sc.nextInt();
10        for(int i = 0 ; i < n ; i++) {
11            for(int j = 0 ; j < i+1 ; j++)
12                System.out.print("*");
13            System.out.println();
14        }
15    }
16}
```

Part – C

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows numerous Java files under the assignment2 project, including Distance.java, Employee.java, EmployeeFinalizer.java, Employee.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestmt.java, UncheckedException.java, WrapperClass.java, and several assignment2\_041 files.
- Code Editor:** Displays the content of StarPattern3.java:

```
System.out.println("Enter the number of rows : ");
int n = sc.nextInt();
for(int i = 0 ; i <= n ; i++) {
    for(int j = 0 ; j < i ; j++) {
        System.out.print(" ");
    }
    for(int k = i ; k < (2*n-(i+1)) ; k++) {
        System.out.print("*");
    }
    System.out.println();
}
```
- Console:** Shows the output of the program when run with the input "5":

```
Enter the number of rows :
5
*****
 ****
  ***
   *

```
- System Tray:** Shows system information including temperature (34°C), battery level (Haze), and date/time (22-Apr-2022, 11:12:15 am).

**Q3. Write a Java program to Print Right Triangle Number Pattern Type**

10987  
456  
32  
1

**Code:**

```
package assignment2;
import java.util.Scanner;
public class RightTriangleStar {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n,a;
        System.out.println("Enter a Number : ");
        n = sc.nextInt();
        for(int i=n;i>=1;i--) {
            if (i!=1) {
                if(i%2==0) {
                    a=(i*(i+1))/2;
                    for(int j=1;j<=i;j++){
                        System.out.print(a--);
                    }
                }
                else{
                    a=(i*(i+1))/(2+1);
                    for(int j=1;j<=i;j++) {
                        System.out.print(a++);
                    }
                }
            System.out.println();
        }
        Else {
            System.out.print(i);
        }
    }
    sc.close();
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the workspace structure with various Java files under assignment1 and assignment2.
- Code Editor:** Displays the `RightTriangleStar.java` file content:1 package assignment2;
2
3 import java.util.Scanner;
4
5 public class RightTriangleStar {
6 public static void main(String[] args)
7 {
8 Scanner sc = new Scanner(System.in);
9 int n,a;
10 System.out.println("Enter a Number : ");
11 n = sc.nextInt();
12
13 for(a=1;a<=n;a++)
14 {
15 for(int i=1;i<=a;i++)
16 {
17 System.out.print("\*");
18 }
19 System.out.println();
20 }
21 }
22}
- Console Output:** Shows the execution of the program and its output:

```
4
Enter a Number :
10987
456
32
1
```
- System Tray:** Shows system information like temperature (34°C), battery level, and date/time (22-Apr-2022, 11:22).

**Q4. Write a program to Check if the given string is Palindrome or not.**

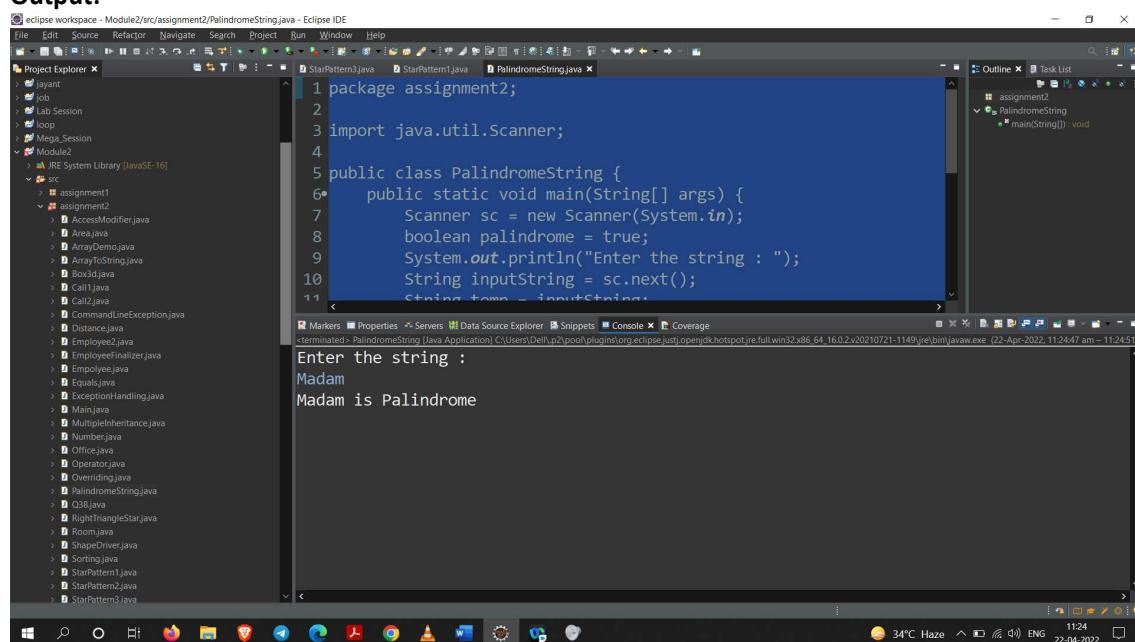
**Code:**

```
package assignment2;

import java.util.Scanner;

public class PalindromeString {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        boolean palindrome = true;
        System.out.println("Enter the string : ");
        String inputString = sc.next();
        String temp = inputString;
        inputString = inputString.toLowerCase();
        for(int i=0, j=inputString.length()-1; i <= inputString.length()/2 ; i++, j--) {
            if(inputString.charAt(i) != inputString.charAt(j)) {
                palindrome = false;
            }
        }
        if(palindrome)
            System.out.println(temp+" is Palindrome");
        else
            System.out.println(temp+" is not Palindrome");
        sc.close();
    }
}
```

**Output:**



**Q5. Write a JAVA program for computing sum of two integers and floats using abstract classes.**

**Code:**

```
package assignment2;

abstract class sum{
    public abstract int integer(int a, int b);
    public abstract float floating(float a, float b);
}

public class SumUsingAbstract extends sum{
    public int integer(int a, int b) {
        int sum = a + b;
        return sum;
    }
    public float floating(float a, float b) {
        float sum = a + b;
        return sum;
    }
    public static void main(String[] args) {
        SumUsingAbstract s = new SumUsingAbstract();
        System.out.println(s.integer(10,20));
        System.out.println(s.floating(30.4f,90.2f));
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Distance.java, Employee.java, EmployeeInitializer.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBMM.java, StringDemo.java, StudentMarks.java, SumArray.java, SumOfWhile.java, SumOfObject.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, and WrappingClass.java.
- SumUsingAbstract.java Content:**

```
public float floating(float a, float b) {
    float sum = a + b;
    return sum;
}
public static void main(String[] args) {
    SumUsingAbstract s = new SumUsingAbstract();
    System.out.println(s.integer(10,20));
    System.out.println(s.floating(30.4f,90.2f));
}
```
- Console View:** Displays the output of the program:  
30  
120.6
- System Status:** Shows the system temperature (34°C), battery level (Haze), and date/time (22-Apr-2022, 11:30:18 am - 11:30:19).

**Q6. Write a program to give the examples of operators.**

- 1) Increment and decrement operators.
- 2) Bitwise Complement Operator.
- 3) Arithmetic operator.
- 4) Relational Operator
- 5) Bitwise operator.
- 6) Conditional Operator

Code:

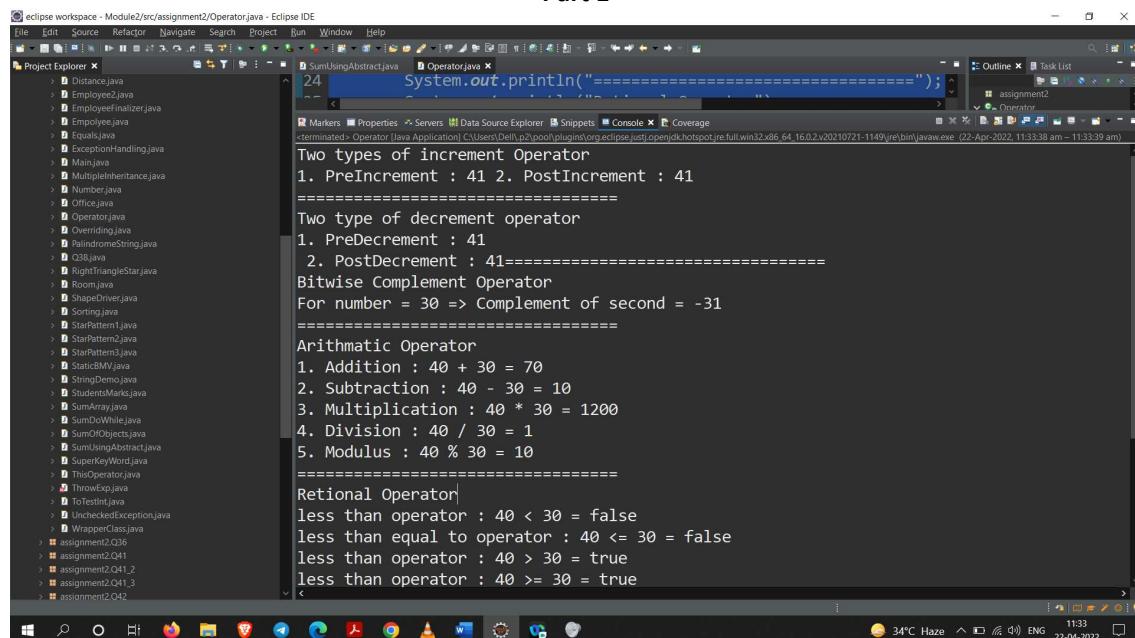
```
package assignment2;

public class Operator {
    public static void main(String[] args) {
        int first = 40, second = 30;
        String str = "abc";
        System.out.println("Two types of increment Operator");
        System.out.print("1. PreIncrement : "+(++first));
        System.out.println(" 2. PostIncrement : "+(first++));
        System.out.println("=====");
        System.out.println("Two type of decrement operator");
        System.out.println("1. PreDecrement : "+(--first));
        System.out.print(" 2. PostDecrement : "+(first--));
        System.out.println("=====");
        System.out.println("Bitwise Complement Operator");
        System.out.println("For number = "+second+" => Complement of
second = "+~second);
        System.out.println("=====");
        System.out.println("Arithmatic Operator");
        System.out.println("1. Addition : "+(first)+" + "+(second)+" =
+(first+second));
        System.out.println("2. Subtraction : "+(first)+" -
"+(second)+" = "+(first-second));
        System.out.println("3. Multiplication : "+(first)+" *
"+(second)+" = "+(first*second));
        System.out.println("4. Division : "+(first)+" / "+(second)+" =
+(first/second));
        System.out.println("5. Modulus : "+(first)+" % "+(second)+" =
+(first%second));
        System.out.println("=====");
        System.out.println("Relational Operator");
        System.out.println("less than operator : "+(first)+" <
"+(second)+" = "+(first<second));
        System.out.println("less than equal to operator :
"+(first)+" <= "+(second)+" = "+(first<=second));
        System.out.println("less than operator : "+(first)+" >
"+(second)+" = "+(first>second));
        System.out.println("less than operator : "+(first)+" >=
"+(second)+" = "+(first>=second));
        System.out.println("str instanceof String : "+str instanceof
String);
        System.out.println("=====");
```

```
System.out.println("Relational Operator");
    System.out.println("Relational & "+(first)+" & "+(second)+"
= "+(first&second));
    System.out.println("Relational | "+(first)+" | "+(second)+"
= "+(first|second));
    System.out.println("Relational ^ "+(first)+" ^ "+(second)+"
= "+(first^second));
    System.out.println("=====");
    System.out.println("Conditional Operator");
    System.out.println("first<second:first?second ==> "+(first <
second ? first : second));
}
}
```

### Output:

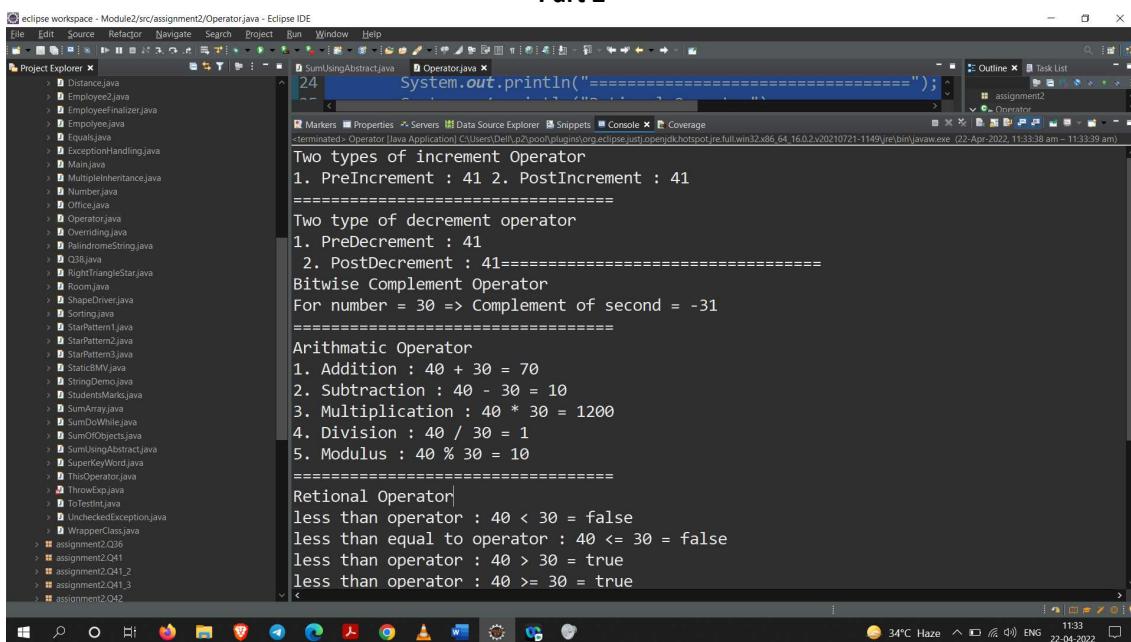
#### Part 1



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files including Distance.java, Employee2.java, EmployeeFinalizer.java, Employee.java, Equals.java, ExceptionHandling.java, Main.java, MultipInheritance.java, Number.java, Palindrome.java, Operator.java, Overriding.java, PalindromeString.java, Q3.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBMMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, and WrapperClass.java.
- Operator.java Content:** The code for the Operator class is displayed in the editor.
- Console Output:** The output window shows the execution of the program:
  - Two types of increment Operator
  - 1. PreIncrement : 41 2. PostIncrement : 41
  - =====
  - Two type of decrement operator
  - 1. PreDecrement : 41
  - 2. PostDecrement : 41
  - =====
  - Bitwise Complement Operator
  - For number = 30 => Complement of second = -31
  - =====
  - Arithmetic Operator
  - 1. Addition : 40 + 30 = 70
  - 2. Subtraction : 40 - 30 = 10
  - 3. Multiplication : 40 \* 30 = 1200
  - 4. Division : 40 / 30 = 1
  - 5. Modulus : 40 % 30 = 10
  - =====
  - Relational Operator
  - less than operator : 40 < 30 = false
  - less than equal to operator : 40 <= 30 = false
  - less than operator : 40 > 30 = true
  - less than operator : 40 >= 30 = true
- System Bar:** Shows system information like temperature (34°C), battery level (Haze), and date/time (22-04-2022 11:33:39 am).

## Part 2



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows numerous Java files including Distance.java, Employee.java, EmployeeFinalizer.java, Employee.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java, UncheckedException.java, WrapperClass.java, and assignment2.Q36, assignment2.Q41, assignment2.Q41\_2, assignment2.Q41\_3, assignment2.Q42.
- Editor:** Displays the code for Operator.java:

```
System.out.println("====");
```

Two types of increment Operator  
1. PreIncrement : 41 2. PostIncrement : 41  
=====

Two type of decrement operator  
1. PreDecrement : 41  
2. PostDecrement : 41=====

Bitwise Complement Operator  
For number = 30 => Complement of second = -31  
=====

Arithmetic Operator  
1. Addition : 40 + 30 = 70  
2. Subtraction : 40 - 30 = 10  
3. Multiplication : 40 \* 30 = 1200  
4. Division : 40 / 30 = 1  
5. Modulus : 40 % 30 = 10  
=====

Relational Operator  
less than operator : 40 < 30 = false  
less than equal to operator : 40 <= 30 = false  
less than operator : 40 > 30 = true  
less than operator : 40 >= 30 = true
- OS Taskbar:** Shows system icons for search, file explorer, task manager, and network, along with weather (34°C Haze), battery (11:33), and date (22-04-2022).

**Q7. Write a JAVA program which computes sum of two objects by accepting the data from command prompt.**

**Code:**

```
package assignment2;

import java.util.Scanner;

public class SumOfObjects {
    int a;
    int b;
    static int x;
    static int y;
    SumOfObjects(){}
    SumOfObjects(int a, int b){
        this.a = a;
        this.b = b;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter values of x1 and y1 : ");
        x = sc.nextInt();
        y = sc.nextInt();
        SumOfObjects s1 = new SumOfObjects(x,y);
        System.out.println("Enter values of x2 and y2 : ");
        x = sc.nextInt();
        y = sc.nextInt();
        SumOfObjects s2 = new SumOfObjects(x,y);
        SumOfObjects s3 = new SumOfObjects();
        s3.a = s1.a + s2.a;
        s3.b = s1.b + s2.b;
        System.out.println("sum of x : "+s3.a);
        System.out.println("sum of y : "+s3.b);
        sc.close();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows numerous Java files under the "src" folder, including Distance.java, Employee.java, EmployeeFinalizer.java, Employee.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticDMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java, UncheckedException.java, WrapperClass.java, and assignment2.036 through 042.
- Code Editor:** Displays the `SumOfObjects.java` file with the following code:

```
1 package assignment2;
2
3 import java.util.Scanner;
4
5 public class SumOfObjects {
6     int a;
7 }
```
- Outline View:** Shows the class structure of `SumOfObjects`, including fields `a`, `b`, `c`, `x`, and `y`, and methods `SumOfObjects()`, `SumOfObjects(int, int)`, and `sum(String[])`.
- Console:** Displays the execution output:

```
Enter values of x1 and y1 :
20 50
Enter values of x2 and y2 :
30 60
sum of x : 50
sum of y : 110
```

---

**Q8. Write a JAVA program which illustrates the concept of access specifiers.**

**Code:**

**DEFAULT SPECIFIER**

**Part - A**

```
package assignment2.Q8AccessSpecifier.Default;

public class D1 {
    D1(){
        System.out.println("default constructor Parent");
    }
    int a = 10;
    void test() {
        System.out.println("default method Parent");
    }
}
```

**Part – B**

```
package assignment2.Q8AccessSpecifier.Default;

public class D2 extends D1{
    public static void main(String[] args) {
        D2 d = new D2();
        d.test();
        System.out.println(d.a);
    }
}
```

**Part – C**

```
package assignment2.Q8AccessSpecifier.Default;

public class D3 {
    public static void main(String[] args) {
        D1 d = new D1();
        d.test();
        System.out.println(d.a);
    }
}
```

**Output:**

**From Part – B**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files including Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticCMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, WrappeClass.java, assignments2.Q8AccessSpecifier.Default, assignments2.Q8AccessSpecifier.Private, assignments2.Q8AccessSpecifier.Protected, assignments2.Q8AccessSpecifier.Public, and assignments2.Q8AccessSpecifier.Public.Diff.Package.
- Code Editor:** Displays the Java code for the D2 class:

```
1 package assignment2.Q8AccessSpecifier.Default;
2
3 public class D2 extends D1{
4     public static void main(String[] args) {
5         D2 d = new D2();
6         d.test();
7         System.out.println(d.a);
8     }
9 }
```
- Outline View:** Shows the class D2 and its main() method.
- Console:** Displays the output of the program: **default constructor Parent** and **default method Parent**.
- System Tray:** Shows the date and time as 23-Apr-2022, 02:03, and the system temperature as 34°C Haze.

**From Part – C**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the same set of Java files as in Part B.
- Code Editor:** Displays the Java code for the D3 class:

```
1 package assignment2.Q8AccessSpecifier.Default;
2
3 public class D3 {
4     public static void main(String[] args) {
5         D1 d = new D1();
6         d.test();
7         System.out.println(d.a);
8     }
9 }
```
- Outline View:** Shows the class D3 and its main() method.
- Console:** Displays the output of the program: **default constructor Parent** and **default method Parent**.
- System Tray:** Shows the date and time as 23-Apr-2022, 02:03, and the system temperature as 34°C Haze.

**Conclusion:**

From the Output of Part – B and Part – C, we see that we can access the data having default access modifier in same class not only in the Sub-Class but also in the Non – SubClass.

**Rules:**

For **default** access modifier,

Members: If a member is made “default” then it can be accessed in the same class and anywhere in same package. But then in different package default will not work.

Constructor: If a constructor is made “default” then its object can be created anywhere in a same package but then in a different package its object cannot be created.

Class: If a class is made “default” then it can be used in the same package, but then in a different package it cannot be used.

**PRIVATE SPECIFIER**

**Part - A**

```
package assignment2.Q8AccessSpecifier.Private;

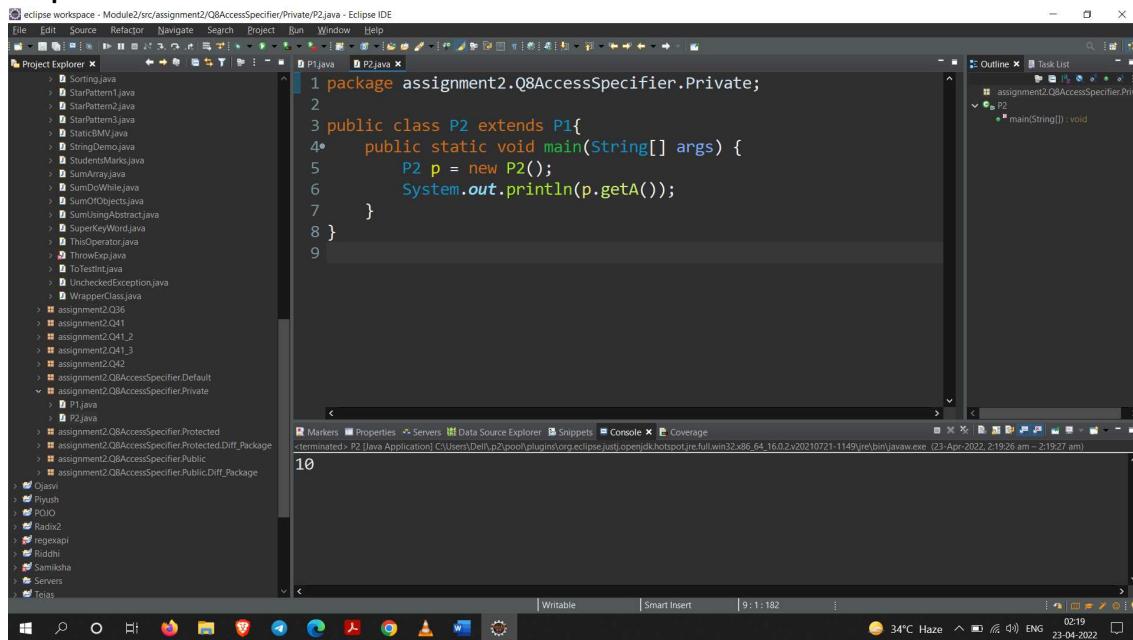
public class P1 {
    private int a = 10;
    public int getA() {
        return a;
    }
}
```

**Part - B**

```
package assignment2.Q8AccessSpecifier.Private;

public class P2 extends P1{
    public static void main(String[] args) {
        P2 p = new P2();
        System.out.println(p.getA());
    }
}
```

### Output:



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files including Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBtMV.java, StringDemo.java, StudentsMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java, UncheckedException.java, WrapperClass.java, assignment2.Q8AccessSpecifier, assignment2.Q8AccessSpecifier.Protected, assignment2.Q8AccessSpecifier.Default, assignment2.Q8AccessSpecifier.Private, assignment2.Q8AccessSpecifier.Public, and P1.java.
- Code Editor:** Displays the code for P2.java:
 

```
1 package assignment2.Q8AccessSpecifier.Private;
2
3 public class P2 extends P1{
4     public static void main(String[] args) {
5         P2 p = new P2();
6         System.out.println(p.getA());
7     }
8 }
```
- Outline View:** Shows the class structure with P2 selected.
- Task List:** Shows a task for P2.
- Console:** Shows the message "terminated-> P2 [Java Application] C:\Users\Delhi\_p2\workspace\assignment2\Q8AccessSpecifier\src\assignment2\Q8AccessSpecifier\Private\P2.java (23-Apr-2022, 2:19:26 am - 2:19:27 am)".
- System Tray:** Shows the date and time as 23-04-2022, 02:19.

### Conclusion:

From the Output of Part – B, we see that we cannot access the data having private access modifier in any class without getters.

### Rules:

For **Private** access modifier,

**Members:** If a member is made private then it can be accessed only in same class.

**Constructor:** If a constructor made private its objects cannot be created in other class. Its object can be created only in same class.

**Class:** Class cannot be made private.

### PROTECTED SPECIFIER

#### Part – A

```
package assignment2.Q8AccessSpecifier.Protected;

public class Protected1 {
    protected int pro = 6;
    protected void test() {
        System.out.println("Test");
    }
}
```

**Part – B**

```
package assignment2.Q8AccessSpecifier.Protected;

public class Protected2 extends Protected1{
    public static void main(String[] args) {
        Protected2 p = new Protected2();
        System.out.println(p.getClass());
        System.out.println(p.pro);
    }
}
```

**Part – C**

```
package assignment2.Q8AccessSpecifier.Protected;
//without inheritance
public class Protected3 {
    public static void main(String[] args) {
        Protected1 p1 = new Protected1();
        System.out.println(p1.getClass());
        p1.test();
        System.out.println(p1.pro);
    }
}
```

**Part – D**

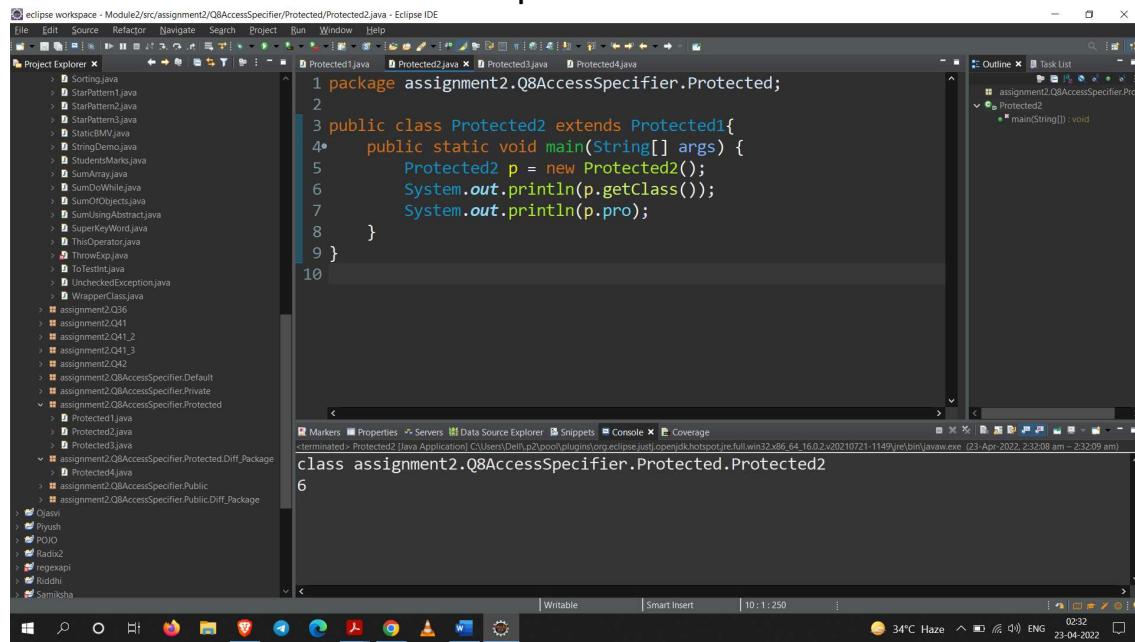
```
package assignment2.Q8AccessSpecifier.Protected.Diff_Package;

import assignment2.Q8AccessSpecifier.Protected.Protected1;

public class Protected4 extends Protected1 {
    public static void main(String[] args) {
        Protected4 p4 = new Protected4();
        System.out.println(p4.getClass());
        p4.test();
        System.out.println(p4.pro);
    }
}
```

**Output:**

**Output of Part - B**



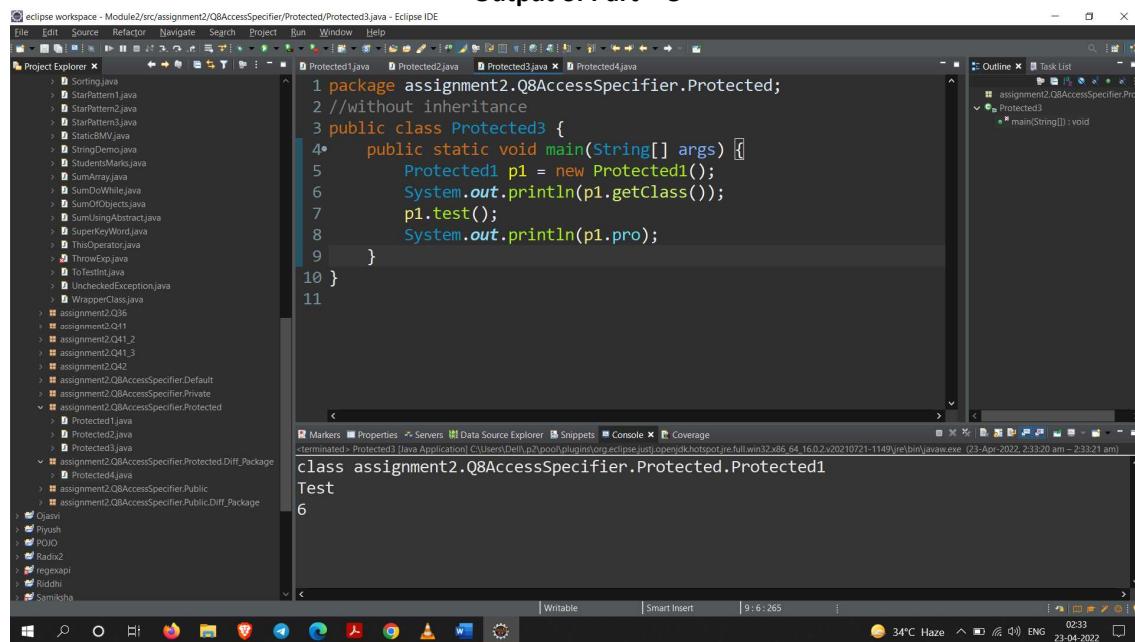
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files including Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticMV.java, StringDemo.java, StudentsMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java, UncheckedException.java, WrapperClass.java, assignment2.Q8AccessSpecifier.Protected, assignment2.Q8AccessSpecifier.Q41, assignment2.Q8AccessSpecifier.Q41\_2, assignment2.Q8AccessSpecifier.Q41\_3, assignment2.Q8AccessSpecifier.Q42, assignment2.Q8AccessSpecifier.Default, assignment2.Q8AccessSpecifier.Private, assignment2.Q8AccessSpecifier.Protected1, Protected2.java, Protected3.java, Protected4.java, assignment2.Q8AccessSpecifier.Protected.Diff.Package, assignment2.Q8AccessSpecifier.Public, assignment2.Q8AccessSpecifier.Public.Diff.Package.
- Code Editor:** Displays the following Java code:

```
1 package assignment2.Q8AccessSpecifier.Protected;
2
3 public class Protected2 extends Protected1{
4     public static void main(String[] args) {
5         Protected2 p = new Protected2();
6         System.out.println(p.getClass());
7         System.out.println(p.pro);
8     }
9 }
10
```

The code editor has tabs for Protected1.java, Protected2.java, Protected3.java, and Protected4.java. The status bar at the bottom shows the date and time as 23-04-2022 02:32.

**Output of Part – C**



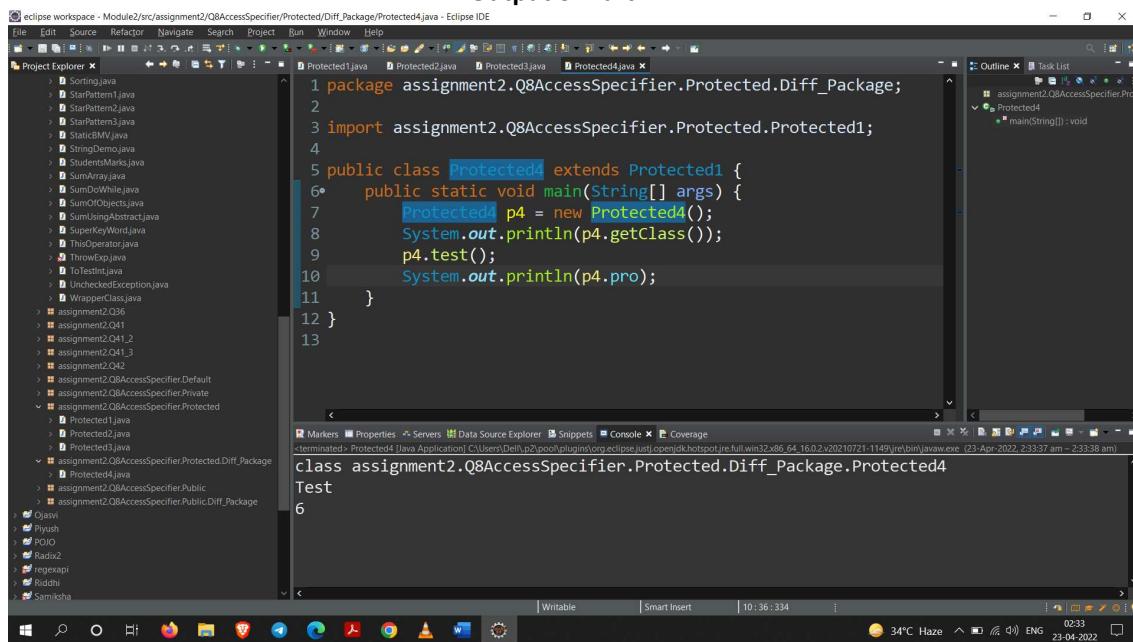
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the same set of Java files as Part B.
- Code Editor:** Displays the following Java code:

```
1 package assignment2.Q8AccessSpecifier.Protected;
2 //without inheritance
3 public class Protected3 {
4     public static void main(String[] args) {
5         Protected1 p1 = new Protected1();
6         System.out.println(p1.getClass());
7         p1.test();
8         System.out.println(p1.pro);
9     }
10 }
11
```

The code editor has tabs for Protected1.java, Protected2.java, Protected3.java, and Protected4.java. The status bar at the bottom shows the date and time as 23-04-2022 02:33.

### Output of Part – D



```

eclipse workspace - Module2/src/assignment2/Q8AccessSpecifier/Protected/Diff_Package/Protected4.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer Task List Outline
1 package assignment2.Q8AccessSpecifier.Protected.Diff_Package;
2
3 import assignment2.Q8AccessSpecifier.Protected.Protected1;
4
5 public class Protected4 extends Protected1 {
6     public static void main(String[] args) {
7         Protected4 p4 = new Protected4();
8         System.out.println(p4.getClass());
9         p4.test();
10        System.out.println(p4.pro);
11    }
12 }
13

Markers Properties Servers Data Source Explorer Snippets Console Coverage
<terminated> - Protected4 [Java Application] C:\Users\Delhi.p2\pgcl\ou\upm\con\ eclipse\jst\openjdk\hotspot\ire\full\win32\x86_64_16.0.2\20210721-1149\bin\javaw.exe (23-Apr-2022 2:33:37 am - 2:33:38 am)
class assignment2.Q8AccessSpecifier.Protected.Diff_Package.Protected4
Test
6

```

#### Conclusion:

From the Output of Part – B, Part – C and Part – D, we see that we can access the data anywhere in same package also in another package only when the class is inherited.

#### Rules:

For **Protected** access modifier,

**Members:** If a member is made protected then it can be accessed in same class and some package and different package only through inheritance.

**Constructor:** If a constructor is made protected then its object can be created only in the same package and cannot create the object in different package.

\*Default and protected are working similar on constructor.

**Class:** Class cannot be made protected.

**PUBLIC SPECIFIER**

**Part – A**

```
package assignment2.Q8AccessSpecifier.Public;

public class Public1 {
    public Public1() {
        System.out.println("Public Members Can Access Anywhere");
    }
    public void test() {
        System.out.println("Public Method");
    }
}
```

**Part – B**

```
package assignment2.Q8AccessSpecifier.Public.Diff_Package;

import assignment2.Q8AccessSpecifier.Public.*;

public class Public2{
    public static void main(String[] args) {
        Public1 p = new Public1();
        System.out.println(p.getClass());
        p.test();
    }
}
```

**Part – C**

```
package assignment2.Q8AccessSpecifier.Public;
//accessing data without inheritance due to public access member
public class Public3 {
    public static void main(String[] args) {
        Public1 p3 = new Public1();
        Public3 p = new Public3();
        System.out.println(p.getClass());
        p3.test();
    }
}
```

**Part – D**

```
package assignment2.Q8AccessSpecifier.Public;

public class Public4 extends Public1{
    public static void main(String[] args) {
        Public4 p4 = new Public4();
        System.out.println(p4.getClass());
        p4.test();
    }
}
```

**Output:**

**Output for Part – B**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files and packages, including `StudentsMarks.java`, `SumArray.java`, `SumDoWhile.java`, `SumOfObjects.java`, `SumUsingAbstract.java`, `SuperKeyWord.java`, `ThisOperator.java`, `ThrowExp.java`, `ToTested.java`, `UncHECKedException.java`, `WrapperClass.java`, `assignment2.Q8AccessSpecifier.Public.Default`, `assignment2.Q8AccessSpecifier.Private`, `assignment2.Q8AccessSpecifier.Protected`, `assignment2.Q8AccessSpecifier.Protected.Diff.Package`, `assignment2.Q8AccessSpecifier.Public`, `Public1.java`, `Public2.java`, `Public3.java`, and `assignment2.Q8AccessSpecifier.Public.Diff.Package`.
- Code Editor:** Displays the `Public2.java` file with the following code:

```
1 package assignment2.Q8AccessSpecifier.Public.Diff.Package;
2
3 import assignment2.Q8AccessSpecifier.Public.*;
4
5 public class Public2{
6     public static void main(String[] args) {
7         Public1 p = new Public1();
8         System.out.println(p.getClass());
9         p.test();
10    }
11 }
```
- Console:** Shows the output of the execution:

```
Public Members Can Access Anywhere
class assignment2.Q8AccessSpecifier.Public.Public1
Public Method
```
- System Tray:** Shows the date and time as 23-04-2022, 03:38:59 am.

**Output for Part – C**

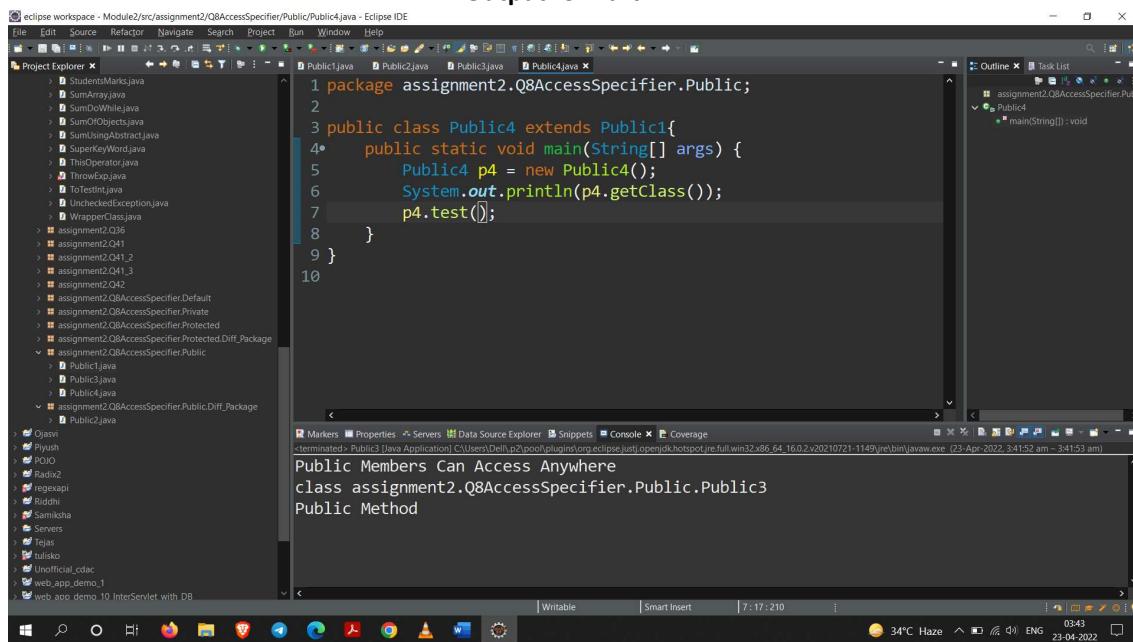
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files and packages, including `StudentsMarks.java`, `SumArray.java`, `SumDoWhile.java`, `SumOfObjects.java`, `SumUsingAbstract.java`, `SuperKeyWord.java`, `ThisOperator.java`, `ThrowExp.java`, `ToTested.java`, `UncHECKedException.java`, `WrapperClass.java`, `assignment2.Q41`, `assignment2.Q41_1`, `assignment2.Q41_2`, `assignment2.Q41_3`, `assignment2.Q42`, `assignment2.Q8AccessSpecifier.Default`, `assignment2.Q8AccessSpecifier.Private`, `assignment2.Q8AccessSpecifier.Protected`, `assignment2.Q8AccessSpecifier.Protected.Diff.Package`, `assignment2.Q8AccessSpecifier.Public`, `Public1.java`, `Public2.java`, `Public3.java`, and `assignment2.Q8AccessSpecifier.Public.Diff.Package`.
- Code Editor:** Displays the `Public3.java` file with the following code:

```
1 package assignment2.Q8AccessSpecifier.Public;
2 //accessing data without inheritance due to public access member
3 public class Public3 {
4     public static void main(String[] args) {
5         Public1 p3 = new Public1();
6         Public1 p = new Public3();
7         System.out.println(p.getClass());
8         p3.test();
9     }
10 }
```
- Console:** Shows the output of the execution:

```
Public Members Can Access Anywhere
class assignment2.Q8AccessSpecifier.Public.Public3
Public Method
```
- System Tray:** Shows the date and time as 23-04-2022, 03:41:52 am.

### Output for Part – D



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows several Java files including `StudentsMarks.java`, `SumArray.java`, `SumDoWhile.java`, `SumOfObjects.java`, `SumUsingAbstract.java`, `SuperKeyWord.java`, `ThisOperator.java`, `ThrowExp.java`, `ToTest.java`, `UncheckedException.java`, `WrapperClass.java`, and multiple files under `assignment2.Q8AccessSpecifier` and `Public` packages.
- Code Editor:** Displays the following Java code in `Public4.java`:
 

```
1 package assignment2.Q8AccessSpecifier.Public;
2
3 public class Public4 extends Public1{
4     public static void main(String[] args) {
5         Public4 p4 = new Public4();
6         System.out.println(p4.getClass());
7         p4.test();
8     }
9 }
10
```
- Console:** Shows the output of the executed code:
 

```
Public Members Can Access Anywhere
class assignment2.Q8AccessSpecifier.Public.Public4
Public Method
```
- System Tray:** Shows system information like temperature (34°C), battery level, and network status.

#### Conclusion:

From the Output of Part – B, Part – C and Part – D, we see that we can access the data anywhere in same package also in another package.

#### Rules:

For **Private** access modifier,  
**Members:** Public members can be accessed in same class, in same package as well as in different package. It gives us maximum visibility.  
**Constructor:** If constructor is made public then its objects can be created in any package.  
**Class:** If you make a class public then it can be used anywhere in any package

**Q9. Write a program to print the array as a string.**

**Code:**

```
package assignment2;

import java.util.Scanner;

public class ArrayToString {
    @SuppressWarnings("unused")
    public static void main(String[] args) {
        //ArrayToString a = new ArrayToString();
        char c[] = new char[10];
        String outputString = "";
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter 10 characters : ");
        int i = 0;
        for(int j : c) {
            c[i] = sc.next().charAt(0);
            i++;
        }
        for(char k : c)
            outputString += k;
        System.out.println("Output : "+outputString);
        sc.close();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like assignment1, assignment2, and various Java files such as AccessModifier.java, Area.java, Box3d.java, Call1.java, Call2.java, CommandLineException.java, Distance.java, Employee1.java, Employee2.java, EmployeeFinalizer.java, Employee.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, and ShapeDriver.java.
- Code Editor:** Displays the Java code for `ArrayToString.java`. The code defines a class `ArrayToString` with a `main` method that reads 10 characters from the user and prints them as a string.
- Console:** Shows the execution output:

```
Enter 10 characters :
s a r a n g d e o d
Output : sarangdeod
```
- Bottom Status Bar:** Shows system information including temperature (34°C), battery level (Haze), and date/time (23-04-2022).

**Q10. Write a do-while loop that asks the user to enter two numbers. The numbers should be added and the sum displayed. The loop should ask the user whether he or she wishes to perform the operation again. If so, the loop should repeat; otherwise, it should terminate.**

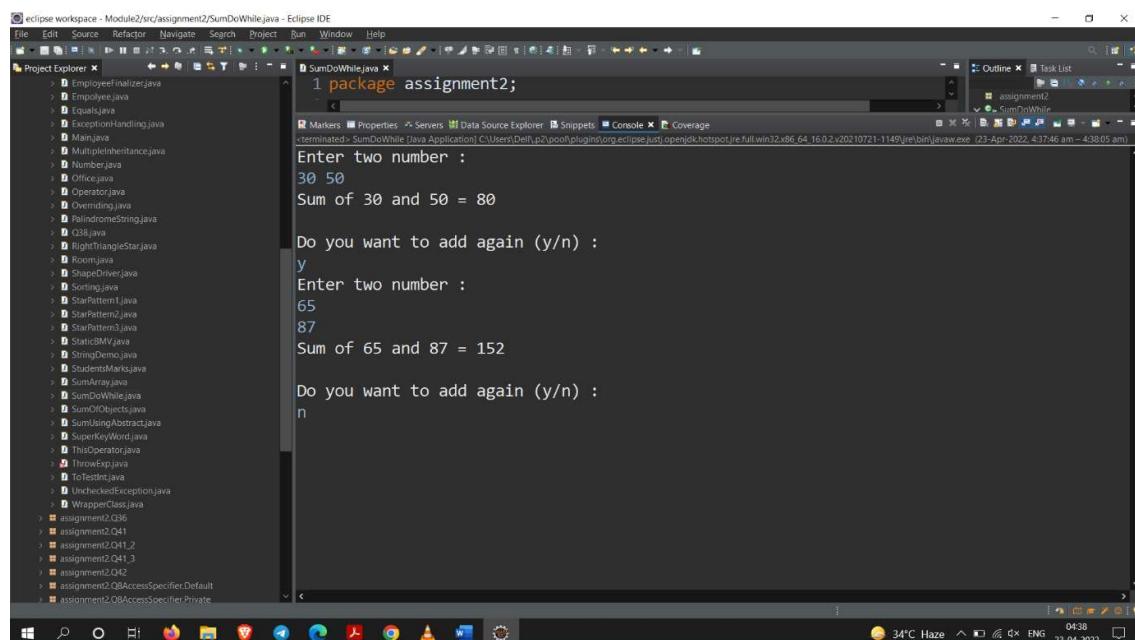
**Code:**

```
package assignment2;

import java.util.Scanner;

public class SumDoWhile {
    public static char repeat = 'n';
    public static void main(String[] args) {
        int sum = 0;
        int first = 0;
        int second = 0;
        Scanner sc = new Scanner(System.in);
        do {
            System.out.println("Enter two number : ");
            first = sc.nextInt();
            second = sc.nextInt();
            sum = first + second;
            System.out.println("Sum of "+first+" and "+second+" = "+sum);
            System.out.println("\nDo you want to add again (y/n) : ");
            repeat = sc.next().charAt(0);
            repeat = Character.toLowerCase(repeat);
            if(repeat != 'n' && repeat != 'y') {
                System.out.println("Invalid input");
                System.out.println("Program terminating");
            }
        }while(repeat=='y');
        sc.close();
    }
}
```

**Output:**



**Q11. Write a program to demonstrate static variables, methods, and blocks.**

**Code:**

```
package assignment2;

public class StaticBMV {
    public static void staticMethod() {
        System.out.println("From static Method");
    }
    static
    {
        a=20;
        System.out.println("static block");
    }
    public static int a;
    public static void main(String[] args) {
        System.out.println(a);
        staticMethod();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Distance.java, Employee.java, EmployeeFinalizer.java, Employee2.java, Equals.java, ExceptionHandling.java, Main.java, MultipInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q3.java, RightAngleStar.java, StackDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, WrapperClass.java, and assignment2.Q36, assignment2.Q41, assignment2.Q41\_2, assignment2.Q41\_3.
- Code Editor:** Displays the code for `StaticBMV.java`. The code defines a static variable `a` and a static method `staticMethod()`. The static block prints "static block" and the static method prints "From static Method".
- Outline View:** Shows the class structure with `assignment2` containing `StaticBMV`, which has a static method `staticMethod()` and a static variable `a`.
- Console:** Shows the output of the program:

```
static block
20
From static Method
```
- System Tray:** Shows the date and time as 23-Apr-2022, 4:40:49 am, and the temperature as 34°C Haze.

**Q12. Write a program to give the example for method overriding concepts.**

**Code:**

```
package assignment2;
class Test2{
    public void loading() {
        System.out.println("Parent");
    }
    public void loading2() {
        System.out.println("Parent class");
    }
}
public class Overriding extends Test2{
    @Override
    public void loading() {
        System.out.println("Child");
    }
    public static void main(String[] args) {
        Overriding t = new Overriding();
        t.loading();
        t.loading2();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the Overriding.java file open in the editor. The code is identical to the one provided above. In the bottom right corner of the editor, the output of the program is displayed in the console:

```
Child
Parent class
```

---

**Q13.** Write a program to create a class named shape. In this class we have three sub classes circle, triangle and square each class has two-member function named draw () and erase (). Create these using polymorphism concepts.

Code:

```
package assignment2;
class Shape{
    public void draw() {System.out.println("Universal Draw");}
    public void erase() {System.out.println("Universal erase");}
}
class Circle extends Shape{
    @Override
    public void draw() {
        System.out.println("Circle Draw");
    }
    @Override
    public void erase() {
        System.out.println("Circle erase");
    }
}
class Triangle extends Shape{
    @Override
    public void draw() {
        System.out.println("Triangle Draw");
    }
    @Override
    public void erase() {
        System.out.println("Triangle erase");
    }
}
class Square extends Shape{
    @Override
    public void draw() {
        System.out.println("Square Draw");
    }
    @Override
    public void erase() {
        System.out.println("Square erase");
    }
}
public class ShapeDriver {
    public static void main(String[] args) {
        Circle c = new Circle();
        Triangle t = new Triangle();
        Square s = new Square();
        c.draw();
        c.erase();
        t.draw();
        t.erase();
        s.draw();
        s.erase();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows numerous Java files in the workspace, including Assignment2, Shape, Circle, Triangle, Square, and ShapeDriver.
- ShapeDriver.java Content:** The code defines a `Shape` class with `draw()` and `erase()` methods, and extends it to `Circle`, `Triangle`, and `Square`. It overrides the methods to print specific messages.
- Console Output:** The output window displays the results of running the program:

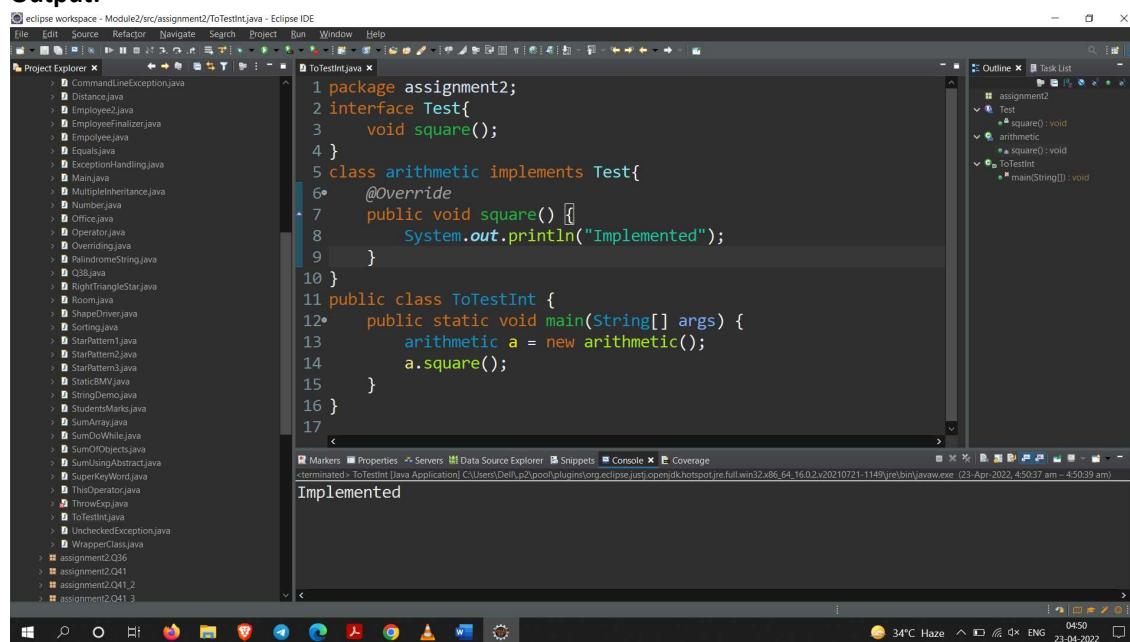
```
Triangle Draw
Triangle erase
Square Draw
Square erase
```
- System Bar:** Shows the date (23-Apr-2022), time (04:47), and weather (34°C Haze).

**Q14.** Write a program to create interface named test. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called ToTestInt in this class use the object of arithmetic class.

Code:

```
package assignment2;
interface Test{
    void square();
}
class arithmetic implements Test{
    @Override
    public void square() {
        System.out.println("Implemented");
    }
}
public class ToTestInt {
    public static void main(String[] args) {
        arithmetic a = new arithmetic();
        a.square();
    }
}
```

**Output:**



**Q15.** Create class Number with only one private instance variable as a double primitive type. To include the following methods (include respective constructors) isZero( ), isPositive(), isNegative( ), isOdd( ), isEven( ), isPrime(), isAmstrong() the above methods return boolean primitive type. getFactorial(), getSqrt(), getSqr(), sumDigits(), getReverse() the above methods return double primitive type. void listFactor(), void dispBinary().

Code:

```
package assignment2;

public class Number {
    private double Var;
    private Number(double var){ this.Var = var; }
    public boolean isZero() {
        if(Var == 0)
            return true;
        return false;
    }
    public boolean isPositive() {
        if(Var > 0)
            return true;
        return false;
    }
    public boolean isNegative(){
        if(Var < 0)
            return true;
        return false;
    }
    public boolean isOdd() {
        if(Var % 2 != 0)
            return true;
        return false;
    }
    public boolean isEven() {
        if(Var % 2 == 0)
            return true;
        return false;
    }
    public boolean isPrime() {
        int count=0;
        if(Var == 1)
            return false;
        for(int i = 2 ; i <= Math.sqrt(Var) ; i++) {
            if(Var % i == 0) {
                count++;
                break;
            }
        }
        if(count == 0 || Var == 2)
            return true;
        return false;
    }
}
```

```
public boolean isArmstrong() {
    double cube = 0;
    int temp = (int)this.Var;
    int count=0;
    int res=0;
    while(temp>0) {
        count++;
        temp/=10;
    }
    temp=(int)this.Var;
    while(temp>0) {
        res=temp%10;
        cube += Math.pow(res,count);
        temp/=10;
    }
    if(cube == Var)
        return true;
    return false;
}
public double getFactroial() {
    double factorial = 1;
    int i = 1;
    while(i<=Var) {
        factorial *= i;
        i++;
    }
    return factorial;
}
public double getSqrt() {
    return Math.sqrt(Var);
}
public double getSqr() {
    return Var * Var;
}
public double sumDigits() {
    double sum = 0;
    int var = (int)Var;
    while(var > 0) {
        sum += (var%10);
        var/=10;
    }
    return sum;
}
public double getReverse() {
    double reverse = 0;
    int var = (int)Var;
    while(var > 0) {
        reverse = ((reverse * 10) + (var % 10));
        var/=10;
    }
    return reverse;
}
```

```
    }
    public void listFactor() {
        for(int i = 0 ; i <= Var ; i++)
            if(Var % i == 0)
                System.out.print(i+" ");
    }
    public void dispBinary() {
        int temp = (int)Var;
        int Binary = 0;
        int i = 1;
        while(temp > 0) {
            Binary += i * (temp % 2);
            i *= 10;
            temp /= 2;
        }
        System.out.println(Binary);
    }
    public static void main(String[] args) {
        Number n = new Number(1);
        System.out.println("is "+n.Var+" Zero : "+n.isZero());
        System.out.println("is "+n.Var+" Positive :
"+n.isPositive());
        System.out.println("is "+n.Var+" Negative :
"+n.isNegative());
        System.out.println("is "+n.Var+" Odd : "+n.isOdd());
        System.out.println("is "+n.Var+" Even : "+n.isEven());
        System.out.println("is "+n.Var+" Prime : "+n.isPrime());
        System.out.println("is "+n.Var+" Armstrong :
"+n.isArmstrong());
        System.out.println("=====");
        System.out.println("Factorial of "+n.Var+" is :
"+n.getFactroial());
        System.out.println("Square root of "+n.Var+" is :
"+n.get.Sqrt());
        System.out.println("Square of "+n.Var+" is : "+n.getSqr());
        System.out.println("Sum of Digits in "+n.Var+" is :
"+n.sumDigits());
        System.out.println("Reverse of number "+n.Var+" is :
"+n.getReverse());
        System.out.println("=====");
        System.out.println("Factor of "+n.Var+" is :");
        n.listFactor();
        System.out.print("\nBinary conversion of "+n.Var+" is : ");
        n.dispBinary();
    }
}
```

**Output:**

```
System.out.println("Factorial of "+n.Var+" is : "+n.getFactorial());
is 9.0 Zero : false
is 9.0 Positive : true
is 9.0 Negative : false
is 9.0 Odd : true
is 9.0 Even : false
is 9.0 Prime : false
is 9.0 Armstrong : true
=====
Factorial of 9.0 is : 362880.0
Square root of 9.0 is : 3.0
Square of 9.0 is : 81.0
Sum of Digits in 9.0 is : 9.0
Reverse of number 9.0 is : 9.0
=====
Factor of 9.0 is :
1 3 9
Binary conversion of 9.0 is : 1001
```

**Q16. Create class box and box3d. box3d is extended class of box. The above two classes going to pull fill following requirement:**

- a) Include constructor.
- b) set value of length, breadth, height
- c) Find out area and volume.

Code:

```
package assignment2;
class Box{
    float length;
    float breadth;
    static float area;
    Box(float length, float breadth){
        this.length = length;
        this.breadth = breadth;
    }
    public float area() {
        area = length * breadth;
        return area;
    }
}

public class Box3d extends Box{
    float height;
    static float volume;
    Box3d(float length, float breadth, float height) {
        super(length, breadth);
        this.height = height;
    }
    public float volume() {
        volume = (area() * height);
        return volume;
    }
    public static void main(String[] args) {
        Box3d b = new Box3d(20.2f,15.2f,25.4f);
        b.area();
        System.out.println("Area : "+b.area());
        b.volume();
        System.out.println("Volume : "+volume);
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the workspace structure with projects like Lab Session, loop, Mega\_Session, and Module2. Under Module2, there is a src folder containing assignment1 and assignment2, which contain various Java files such as AccessModifier.java, Areas.java, ArrayDemo.java, ArrayToString.java, Box3d.java, Call1.java, Call2.java, CommandLineException.java, Distance.java, Employee.java, Employeeinitializer.java, Employee.java, Employee.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBMM.java, and StringDemo.java.
- Code Editor:** Displays the contents of Box3d.java. The code defines a class Box with a static float area method and a class Box3d extending it with a height variable and a volume method.
- Outline View:** Shows the class hierarchy and member variables for Box and Box3d.
- Console:** Displays the output of the application, showing "Area : 307.04" and "Volume : 7798.816".
- System Tray:** Shows system status including temperature (34°C), battery level, and network connection.

**Q17. Write a program to create a room class, the attributes of this class is roomno, roomtype, roomarea and ACmachine. In this class the member functions are setData and displaydata.**

**Code:**

```
package assignment2;

public class Room {
    private int roomNo;
    private String roomType;
    private double roomArea;
    private boolean acMachine;

    public void setData(int roomNo, String roomType, double roomArea,
boolean acMachine){
        this.roomNo = roomNo;
        this.roomType = roomType;
        this.roomArea = roomArea;
        this.acMachine = acMachine;
    }
    public void displayData() {
        System.out.println("Room Number : "+roomNo);
        System.out.println("Room Type : "+roomType);
        System.out.println("Room Area : "+roomArea);
        System.out.println("Room have AC : "+acMachine);
        System.out.println("+=====+");
    }
    public static void main(String args[]) {
        Room r = new Room();
        Room r1 = new Room();
        r.setData(1,"single",10.2,false);
        r1.setData(2,"double",20.2,true);
        r.displayData();
        r1.displayData();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace.
- Editor:** Displays the content of `Room.java`. The code defines a class `Room` with attributes `roomNo`, `roomType`, `roomArea`, and `acMachine`. It includes a constructor and a method `setData`.
- Outline:** Shows the class structure and methods of `Room`.
- Console:** Displays the output of the program, which prints two instances of the `Room` class with their respective attributes.
- Bottom Status Bar:** Shows system information like temperature (34°C), battery level (Haze), and date/time (23-04-2022, 05:02).

```
1 package assignment2;
2
3 public class Room {
4     private int roomNo;
5     private String roomType;
6     private double roomArea;
7     private boolean acMachine;
8
9     public void setData(int roomNo, String roomType, double roomA)
10        this.roomNo = roomNo;
+
+=====
11
12     Room Number : 1
13     Room Type : single
14     Room Area : 10.2
15     Room have AC : false
16 +=====
17
18     Room Number : 2
19     Room Type : double
20     Room Area : 20.2
21     Room have AC : true
22 +=====
```

**Q18. Write a program to give the example for ‘this’ operator. And also use the ‘this’ keyword as return statement.**

**Code:**

```
package assignment2;

public class ThisOperator {
    private String name;
    private int roll;
    public ThisOperator Student(String name, int roll) {
        this.name = name;
        this.roll = roll;
        return this;
    }
    public void displayData() {
        System.out.println("Name : "+name);
        System.out.println("Roll : "+roll);
        System.out.println("=====+");
    }
    public static void main(String[] args) {
        ThisOperator t = new ThisOperator();
        ThisOperator t1 = new ThisOperator();
        ThisOperator t2 = new ThisOperator();
        t.Student("Sarang",26);
        t1.Student("Deepak",300);
        t2.Student("ABC", 400);
        t.displayData();
        t1.displayData();
        t2.displayData();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace.
- ThisOperator.java:** The code is as follows:

```
12 System.out.println("Name : "+name);
13 System.out.println("Roll : "+roll);
14 System.out.println("=====");
15 }
16 public static void main(String[] args) {
17     ThisOperator t = new ThisOperator();
18     ThisOperator t1 = new ThisOperator();
19     ThisOperator t2 = new ThisOperator();
20     t.Student("Sarang",26);
```

- Console Output:** Displays the program's output:

```
Name : Sarang
Roll : 26
=====
Name : Deepak
Roll : 300
=====
Name : ABC
Roll : 400
=====
```
- Outline View:** Shows the class structure and methods.
- Bottom Status Bar:** Shows system information like temperature (34°C), battery level (Haze), and date/time (23-04-2022).

**Q19. write a program to find the sum of command line arguments and count the invalid integers entered.**

**Code:**

```
package assignment2;
// argument : 10 20 a 30 b 40 90 c
public class CommandLineException {
    public static void main(String[] args) {
        int invalid = 0;
        float sum = 0;
        for(int i = 0 ; i < args.length ; i++) {
            try {
                sum += Float.parseFloat(args[i]);
            }catch(Exception e) {
                invalid++;
            }
        }
        System.out.println("Sum : "+sum);
        System.out.println("Invalid input : "+invalid);
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with modules like JRE System Library (JavaSE-16), assignment1, and assignment2.
- Code Editor:** Displays the Java code for `CommandLineException.java`.
- Console View:** Shows the execution results:

```
Sum : 190.0
Invalid input : 3
```
- System Status Bar:** Shows system information: 34°C Haze, 05:11, 23-04-2022.

**Q20. Demonstrate the use of 'super' keyword.**

- a) To refer to a member of super class.
- b) To call super class constructor from sub class constructor.

Code:

```
package assignment2;
class Parent{
    int a = 10;
    Parent(){
        System.out.println("Parent Constructor called");
    }
}
public class SuperKeyWord extends Parent{
    SuperKeyWord(){
        super();
        System.out.println("Child Constructor called");
    }
    public void accessVariable() {
        System.out.println("-----");
        System.out.println("Parent member accessed : ");
        System.out.println("Parent class member a : "+super.a);
    }
    public static void main(String[] args) {
        System.out.println("Constructor called : ");
        SuperKeyWord s = new SuperKeyWord();
        s.accessVariable();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including BoxedJava, Call1.java, Call2.java, CommandLineException.java, Distance.java, Employee.java, EmployeeFinalizer.java, Employee2.java, Equals.java, ExceptionHandling.java, Iterable.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, O3.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBVM.java, StringDemo.java, StudentsMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java, UncheckedException.java, and WrapperClass.java.
- Code Editor:** Displays the SuperKeyWord.java code.
- Console Output:** Shows the execution results:

```
Constructor called :
Parent Constructor called
Child Constructor called
-----
Parent member accessed :
Parent class member a : 10
```
- Outline View:** Shows the class hierarchy: assignment2 > Parent > SuperKeyWord.

**Q21. Define a base class person and a derived class employee with single inheritance.**

-Define **SetData()** member functions in each of the class with different signatures to set the data members and demonstrate overloading of member functions.

-Define **GetData()** member functions in each of the class with same signatures to display data and demonstrate overriding of member functions.

Code:

```
package assignment2;
class Person{
    public int age;
    public String name;
    public void setData(int age) {
        this.age = age;
    }
    public void setData(String name) {
        this.name = name;
    }
    public void GetData() {
        System.out.println("Name : "+this.name);
        System.out.println("Age : "+this.age);
    }
}
public class Empolyee extends Person{
    int id = 26;
    @Override
    public void GetData() {
        System.out.println("Id : "+id);
        System.out.println("Name : "+this.name);
        System.out.println("Age : "+this.age);
    }
    public static void main(String[] args) {
        Empolyee e = new Empolyee();
        e.setData(23);
        e.setData("Sarang");
        e.GetData();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a list of Java files in the assignment2 package, including Areas.java, ArrayDemo.java, ArrayToString.java, Box3d.java, Call1.java, Call2.java, CommandLineException.java, Distance.java, Employee.java, EmployeeFinalizer.java, Employees.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern.java, StarPattern2.java, StarPattern3.java, StaticBMMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java.
- Code Editor (Empolyee.java):** Displays the following Java code:

```
12     System.out.println("Name : "+this.name);
13     System.out.println("Age : "+this.age);
14 }
15 }
16 public class Empolyee extends Person{
17     int id = 26;
18     @Override
19     public void GetData() {
20         System.out.println("Id : "+id);
21         System.out.println("Name : "+this.name);
22         System.out.println("Age : "+this.age);
23     }
24     public static void main(String[] args) {
```
- Outline View:** Shows the class hierarchy: Person (age: int, name: String, setData(int), setData(String)), and Empolyee (id: int, GetData(), main(String[])).
- Console Output:** Displays the execution results:

```
Id : 26
Name : Sarang
Age : 23
```
- System Tray:** Shows system information: 34°C Haze, ENG, 05:17, 23-04-2022.

**Q22. Modify program 21 to define a parametrized constructor and finalizer in each class.**

**Demonstrate calling the constructor of the base class from the constructor of the derived class.**

**-Create objects of person and employee classes to show the order of invocation of constructors.**

**Code:**

```

package assignment2;
class Person1{
    private String name;
    private int age;
    Person1(String name,int age){
        System.out.println("constructor of parent class.");
        this.name = name;
        this.age = age;
    }
    public void getData() {
        System.out.println("Name: "+name);
        System.out.println("Age: "+age);
    }
    @Override
    protected void finalize() {
        System.out.println("GC invoked from parent class");
    }
}
public class EmployeeFinalizer extends Person1{
    int id;
    EmployeeFinalizer(String name, int age , int id) {
        super(name,age);
        this.id=id;
        System.out.println("constructor of child class.");
    }
    @Override
    public void getData() {
        super.getData();
        System.out.println("Id: "+id);
    }
    @Override
    protected void finalize() {
        System.out.println("GC invoked from child class");
    }
    public static void main(String[] args) {
        Person1 p1 = new Person1("Sarang", 24);
        p1.getData();
        System.out.println("-----+");

        EmployeeFinalizer ef = new EmployeeFinalizer("Deepak", 19, 26);
        ef.getData();

        p1=null;
        ef=null;
        System.gc();
    }
}

```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files including `Area.java`, `ArrayListException.java`, `Distance.java`, `EmployeeFinalizer.java`, `Employee.java`, `Exception.java`, `ExceptionHandling.java`, `Main.java`, `MultipInheritance.java`, `Number.java`, `Office.java`, `Operator.java`, `Overriding.java`, `PalindromeString.java`, `Q38.java`, `RightTriangleStar.java`, `Room.java`, `ShapeDriver.java`, `Sorting.java`, `StarPattern1.java`, `StarPattern2.java`, `StarPattern3.java`, `StaticBMMV.java`, `StringDemo.java`, `StudentMarks.java`, `SumArray.java`, `SumDoWhile.java`, `SumOfObjects.java`, `SumUsingAbstract.java`, `SuperKeyWord.java`, `ThisOperator.java`, `ThrowExp.java`, and `ToTestInt.java`.
- EmployeeFinalizer.java:** The code defines a `Person` class and an `EmployeeFinalizer` class. It creates an instance of `Person` named `p1` and prints its data. Then it creates an instance of `EmployeeFinalizer` named `ef` and prints its data. Finally, it sets `p1` and `ef` to null.
- Console Output:** The output window shows the following text:

```
constructor of parent class.  
Name: Sarang  
Age: 24  
+-----+  
constructor of parent class.  
constructor of child class.  
Name: Deepak  
Age: 19  
Id: 26  
GC invoked from child class  
GC invoked from parent class
```
- Outline View:** Shows the class hierarchy with `Person` and `EmployeeFinalizer` classes and their respective methods.

**Q23. Define a class of type Distance that has Feet and Inches as members.**

**-Define a function that adds two Distances passed as argument and returns the sum as another Distance object.**

**Code:**

```
package assignment2;

import java.util.Scanner;

public class Distance {
    int feet;
    int inches;
    Scanner sc = new Scanner(System.in);
    public void distanceInput() {
        System.out.println("Enter the feet : ");
        feet = sc.nextInt();
        System.out.println("Enter the inches : ");
        inches = sc.nextInt();
    }
    public Distance addDistances(Distance obj1, Distance obj2) {
        feet = obj1.feet + obj2.feet;
        inches = obj1.inches + obj2.inches;
        return this;
    }
    public void displayDistance() {
        System.out.println("Feet : "+this.feet+" and Inches : "
"+this.inches);
    }
    public static void main(String[] args) {
        Distance obj1 = new Distance();
        Distance obj2 = new Distance();
        obj1.distanceInput();
        obj2.distanceInput();
        Distance obj3 = new Distance();
        obj3 = obj3.addDistances(obj1, obj2);
        obj3.displayDistance();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows numerous Java files in the assignment2 package, including Area.java, ArrayDemo.java, ArrayToString.java, Box3d.java, Call1.java, Call2.java, CommandLineException.java, Distance.java, Employee.java, EmployeeFinalizer.java, Employees.java, Exception.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBMMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestInt.java.
- Distance.java Content:**

```
1 package assignment2;
2
3 import java.util.Scanner;
4
5 public class Distance {
6     int feet;
7     int inches;
8     Scanner sc = new Scanner(System.in);
9     public void distanceInput() {
10         System.out.println("Enter the feet : ");
11         feet = sc.nextInt();
```
- Console Output:**

```
Enter the feet :
5
Enter the inches :
30
Enter the feet :
7
Enter the inches :
40
Feet : 12 and Inches : 70
```
- System Status Bar:** Shows the date (23-Apr-2022), time (5:22:23 am), and weather (34°C Haze).

**Q24. Write a program to calculate the following**

1. Find the length of array.
2. Demonstrate a one-dimensional array.
3. Demonstrate a two-dimensional array.
4. Demonstrate a multi-dimensional array.

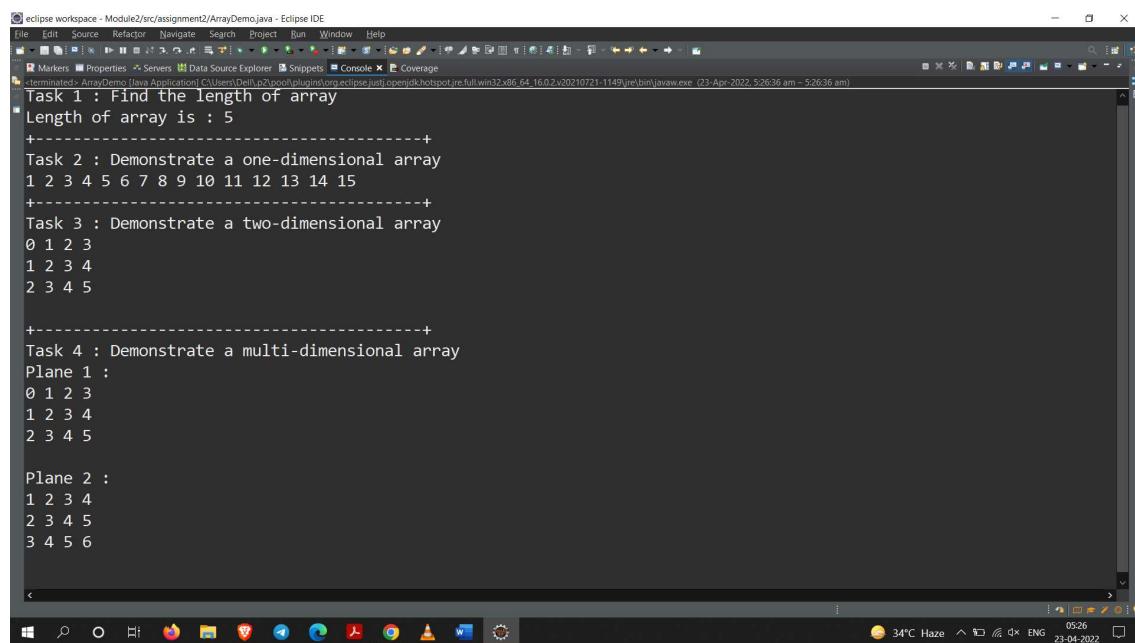
Code:

```
package assignment2;

public class ArrayDemo {
    public static void main(String[] args) {
        System.out.println("Task 1 : Find the length of array");
        int a[] = new int[5];
        System.out.println("Length of array is : "+a.length);
System.out.println("+-----+");
System.out.println("Task 2 : Demonstrate a one-dimensional array");
        a = new int[15];
        //initialize the elements of the 1D array
        for(int i = 0; i < a.length ; i++)
            a[i] = i+1;
        //Display the data in 1D array
        for(int i = 0 ; i < a.length ; i++)
            System.out.print(a[i]+" ");
System.out.println("\n+-----+");
System.out.println("Task 3 : Demonstrate a two-dimensional array");
        int b[][] = new int[3][4];
        for(int i = 0 ;i < b.length;i++) {
            for(int j = 0 ; j < b[0].length;j++) {
                b[i][j]=i+j;
            }
        }
        for(int i=0;i<b.length;i++) {
            for(int j=0;j<b[0].length;j++) {
                System.out.print(b[i][j]+" ");
            }
            System.out.println();
        }
System.out.println("\n+-----+");
System.out.println("Task 4 : Demonstrate a multi-dimensional array");
        int c[][][] = new int[2][3][4];
        for(int i = 0 ;i < c.length ; i++) {
            for(int j = 0 ; j < c[0].length ; j++) {
                for(int k=0 ; k<c[0][0].length ; k++) {
                    c[i][j][k]=i+j+k;
                }
            }
        }
    }
}
```

```
for(int i = 0 ;i < c.length ; i++) {
    System.out.println("Plane "+(i+1)+" : ");
    for(int j = 0 ; j < c[0].length ; j++) {
        for(int k=0 ; k<c[0][0].length ; k++) {
            System.out.print(c[i][j][k]+" ");
        }
        System.out.println();
    }
    System.out.println();
}
```

**Output:**



The screenshot shows the Eclipse IDE interface with a Java application named "ArrayDemo" running. The console tab displays the following output:

```
Task 1 : Find the length of array
Length of array is : 5
+-----+
Task 2 : Demonstrate a one-dimensional array
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
+-----+
Task 3 : Demonstrate a two-dimensional array
0 1 2 3
1 2 3 4
2 3 4 5
+-----+
Task 4 : Demonstrate a multi-dimensional array
Plane 1 :
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

The operating system taskbar at the bottom shows various icons and the date/time: 34°C Haze 05:26 23-04-2022.

**Q25.** Write a program suppose, it is required to build a project consisting of a number of classes, possibly using a large number of programmers. It is necessary to make sure that every class from which all other classes in the project will be inherited. Since any new classes in the project must inherit from the base class, programmers are not free to create a different interface. Therefore, it can be guaranteed that all the classes in the project will respond to the same debugging commands.

Code:

```
package assignment2;
abstract class abs{
    abstract void method();
}
class One extends abs{
    private int a;
    private int b;
    One(int a,int b){
        this.a=a;
        this.b=b;
    }
    @Override
    void method() {
        System.out.println("Class One");
        System.out.println("a = "+a+" b = "+b);
        System.out.println("-----+");
    }
}
class Two extends abs{
    private int a;
    private int b;
    Two(int a, int b){
        this.a = a;
        this.b = b;
    }
    @Override
    void method() {
        System.out.println("Class Two");
        System.out.println("a = "+a+" b = "+b);
        System.out.println("-----+");
    }
}
class Three extends abs{
    private int a;
    private int b;
    Three(int a, int b){
        this.a = a;
        this.b = b;
    }
    @Override
    void method() {
        System.out.println("Class Three");
        System.out.println("a = "+a+" b = "+b);
    }
}
```

```
}

public class Main {
    public static void main(String[] args) {
        One o = new One(10,20);
        Two t = new Two(20,30);
        Three e = new Three(40,50);
        o.method();
        t.method();
        e.method();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace.
- Code Editor:** Displays the Main.java code provided in the question.
- Console Output:** Shows the execution results:

```
Class One
a = 10 b = 20
+-----+
Class Two
a = 20 b = 30
+-----+
Class Three
a = 40 b = 50
```
- OS Taskbar:** Shows system status including temperature (34°C), battery level (Haze), and date/time (23-04-2022).

**Q26. Write a user-defined function to find the sum of an array passed as argument.**

**-Write a program that declares an array of 10 elements and uses this function to**

- a) Find the sum of all elements.
- b) Find the sum of first 5 elements.
- c) Find the sum of last 5 elements.

**Code:**

```
package assignment2;

import java.util.Scanner;

public class SumArray {
    public void sum(int[] a) {
        int add=0;
        for(int x : a)
            add+=x;
        System.out.println("Sum of all elements : "+add);
        add=0;
        for(int i = 0 ; i < 5 ; i++)
            add+=a[i];
        System.out.println("Sum of first five elements : "+add);
        add=0;
        for(int i = a.length-1, count = 0 ; count < 5 ; count++, i--)
            add+=a[i];
        System.out.println("Sum of last five elements : "+add);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = 10;
        int a[] = new int[n];
        System.out.println("Enter "+n+" elements : ");
        for(int i = 0 ; i < n ; i++) {
            a[i] = sc.nextInt();
        }
        SumArray s = new SumArray();
        s.sum(a);
        sc.close();
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows numerous Java files under the assignment2 package, including BoxId.java, Call1.java, Call2.java, CommandLineException.java, Distance.java, Employee.java, Employeeinitializer.java, Employee.java, Equality.java, ExceptionHandling.java, Main.java, MultiplexInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q3.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticBmV.java, StringDemo.java, StudentsMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTestinf.java, UncheckedException.java, and WrapperClass.java.
- Code Editor:** Displays the `SumArray.java` file with the following code:

```
19 }
20 public static void main(String[] args) {
21     Scanner sc = new Scanner(System.in);
22
23     System.out.println("Enter 10 elements : ");
24     for (int i = 0; i < 10; i++) {
25         int num = sc.nextInt();
26         sum += num;
27     }
28
29     System.out.println("Sum of all elements : " + sum);
30     System.out.println("Sum of first five elements : " + sum);
31     System.out.println("Sum of last five elements : " + sum);
32 }
```
- Console Output:** Shows the execution results:

```
Enter 10 elements :
12
65
32
94
25
23
23
10
25
13
Sum of all elements : 322
Sum of first five elements : 228
Sum of last five elements : 94
```
- System Tray:** Shows system information: 34°C Haze, 05:34, ENG, 23-04-2022.

---

**Q27. Create a class named 'Member' having the following members:**

**Data members:**

- 1 - Name
- 2 - Age
- 3 - Phone number
- 4 - Address
- 5 - Salary

It also has a method named 'printSalary' which prints the salary of the members.

Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

**Code:**

```
package assignment2;

class Member {
    String name;
    int age;
    String phoneNumber;
    String address;
    float salary;
    public void printSalary() {
        System.out.println("Salary : "+salary);
    }
    Member(String name,int age,String phoneNumber,String address,float salary){
        this.name=name;
        this.age=age;
        this.phoneNumber=phoneNumber;
        this.address=address;
        this.salary=salary;
    }
}
class Employee extends Member{
    String specialization = "J2EE";
    Employee(String name,int age,String phoneNumber,String address,float salary){
        super(name,age,phoneNumber,address,salary);
    }
    public void printDetails() {
        System.out.println("Name : "+name);
        System.out.println("Age : "+age);
        System.out.println("Phone Number : "+phoneNumber);
        System.out.println("Address : "+address);
        printSalary();
        System.out.println("Specialization : "+specialization);
    }
}
```

```
class Manager extends Member{
    String department = "Designing";
    Manager(String name,int age,String phoneNumber,String address,float salary){
        super(name,age,phoneNumber,address,salary);
    }
    public void printDetails() {
        System.out.println("Name : "+name);
        System.out.println("Age : "+age);
        System.out.println("Phone Number : "+phoneNumber);
        System.out.println("Address : "+address);
        printSalary();
        System.out.println("Department : "+department);
    }
}
public class Office{
    public static void main(String[] args) {
        Employee e = new Employee("Sarang", 24, "9420945517",
        "Akola", 1000000);
        Manager m = new Manager("Deepak", 25, "9420945516", "Akola",
        2000000);
        e.printDetails();
        System.out.println("=====+");
        m.printDetails();
        System.out.println("=====+");
    }
}
```

### Output:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace.
- Code Editor:** Displays the `Office.java` file content.
- Outline View:** Shows the class hierarchy and member variables for `Member` and `Manager`.
- Console Output:** Displays the program's output, showing two sets of details for employees Sarang and Deepak, separated by a double-line separator.

```
1 package assignment2;
2
3 class Member {
4     String name;
5     int age;
6     String phoneNumber;
```

```
Name : Sarang
Age : 24
Phone Number : 9420945517
Address : Akola
Salary : 1000000.0
Specialization : J2EE
=====
Name : Deepak
Age : 25
Phone Number : 9420945516
Address : Akola
Salary : 2000000.0
Department : Designing
=====
```

**Q28.** We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students.

Code:

```

package assignment2;
abstract class Marks { public abstract float getPercentage(); }
class A extends Marks{
    float percentage = 0;
    float m1;
    float m2;
    float m3;
    A(float m1, float m2, float m3){
        this.m1 = m1;
        this.m2 = m2;
        this.m3 = m3;
    }
    @Override
    public float getPercentage() { return ((m1+m2+m3)/3); }
}
class B extends Marks{
    float percentage = 0;
    float m1;
    float m2;
    float m3;
    float m4;
    B(float m1, float m2, float m3, float m4){
        this.m1 = m1;
        this.m2 = m2;
        this.m3 = m3;
        this.m4 = m4;
    }
    @Override
    public float getPercentage() { return ((m1+m2+m3+m4)/4); }
}
public class StudentsMarks {
    public static void main(String[] args) {
        A a1 = new A(90.4f,32.60f,45.21f);
        B b1 = new B(45.26f,97.3f,45.23f,99.36f);
        System.out.println("Percentage of class A : "
        "+a1.getPercentage());
        System.out.println("Percentage of class B : "
        "+b1.getPercentage());
    }
}

```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace.
- Code Editor:** Displays the `StudentsMarks.java` file containing the following code:

```
1 package assignment2;
2 abstract class Marks {
3     public abstract float getPercentage();
4 }
5 class A extends Marks{
6     float percentage = 0;
7     float m1;
8     float m2;
9     float m3;
10    A(float m1, float m2, float m3){
11        this.m1 = m1;
12        this.m2 = m2;
13        this.m3 = m3;
14    }
15    @Override
```

- Outline View:** Shows the class hierarchy and member variables.
- Console:** Displays the output of the program:  
Percentage of class A : 56.06996  
Percentage of class B : 71.7875
- System Tray:** Shows weather information (34°C Haze) and system status.

**Q29.** We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.

**Code:**

```
package assignment2;
abstract class Shapee{
    public abstract void RectangleArea(float length, float breadth);
    public abstract void SquareArea(float side);
    public abstract void CircleArea(float radius);
}
public class Area extends Shapee {

    @Override
    public void RectangleArea(float length, float breadth) {
        System.out.println("Area of Rectangle : "
"+(length*breadth));
    }

    @Override
    public void SquareArea(float side) {
        System.out.println("Area of Square : "+(side*side));
    }

    @Override
    public void CircleArea(float radius) {
        System.out.println("Area of radius : "
"+(3.14*radius*radius));
    }

    public static void main(String[] args) {
        Area a = new Area();
        a.RectangleArea(10.2f, 30.1f);
        a.SquareArea(1.2f);
        a.CircleArea(10.0f);
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the workspace structure with "Module2" selected. Inside "Module2", there is a "src" folder containing numerous Java files such as "assignment1", "Assignment2", "AccessModifier.java", "Area.java", "ArrayDemo.java", "BoxId.java", "Call1.java", "Call2.java", "CommandLineException.java", "Employee.java", "EmployeeFinalizer.java", "Employee.java", "Equals.java", "ExceptionHandling.java", "Main.java", "MultipInheritance.java", "Number.java", "Office.java", "Operator.java", "Overriding.java", "PalindromeString.java", "Q3.java", "RightTriangleStar.java", "Room.java", "ShapeDriver.java", "Sorting.java", "StarPattern1.java", "StarPattern2.java", "StarPattern3.java", "StaticBMMV.java", "StringDemo.java", "StudentsMarks.java", "SumArray.java", and "SumDoWhile.java".
- Code Editor:** Displays the "Area.java" file with the following code:

```
16     System.out.println("Area of Square : "+(side*side));
17 }
18
19* @Override
20 public void CircleArea(float radius) {
21     System.out.println("Area of radius : "+(3.14*radius*radius));
22 }
23
24* public static void main(String[] args) {
25     Area a = new Area();
26     a.RectangleArea(10.2f, 30.1f);
27     a.SquareArea(12.0f);
28     a.CircleArea(10.0f);
29 }
30 }
```
- Outline View:** Shows the class hierarchy and methods defined in "Area.java". It includes "Shape" and "Area" classes with their respective methods: "RectangleArea", "SquareArea", and "CircleArea".
- Console Output:** Shows the terminal output of the program:

```
Area of Rectangle : 307.0
Area of Square : 144.0
Area of radius : 314.0
```
- System Tray:** Shows the Windows taskbar with icons for File Explorer, Task View, Start, Taskbar settings, and other system icons.

---

**Q30. Write a program for the following**

1. Example for call by value.
2. Example for call by reference.

**Code:**

**Part – A**  
**Call by Value**

```
package assignment2;
//Call by value
public class Call1 {
    int a = 30;
    public void setA(int a) {
        a = a + 30;
    }
    public static void main(String[] args) {
        Call1 c = new Call1();
        System.out.println("Before Change : "+c.a);
        c.setA(20);
        System.out.println("After Change : "+c.a);
    }
}
```

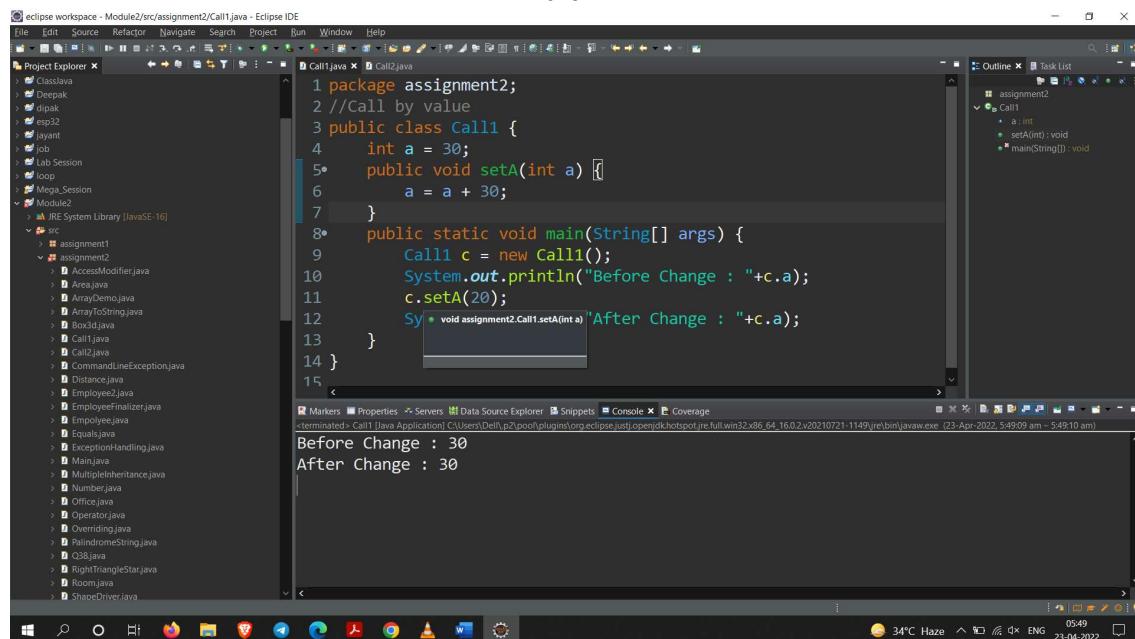
**Part – B**  
**Call by Reference**

```
package assignment2;
//Call by value
public class Call1 {
    int a = 30;
    public void setA(int a) {
        a = a + 30;
    }
    public static void main(String[] args) {
        Call1 c = new Call1();
        System.out.println("Before Change : "+c.a);
        c.setA(20);
        System.out.println("After Change : "+c.a);
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

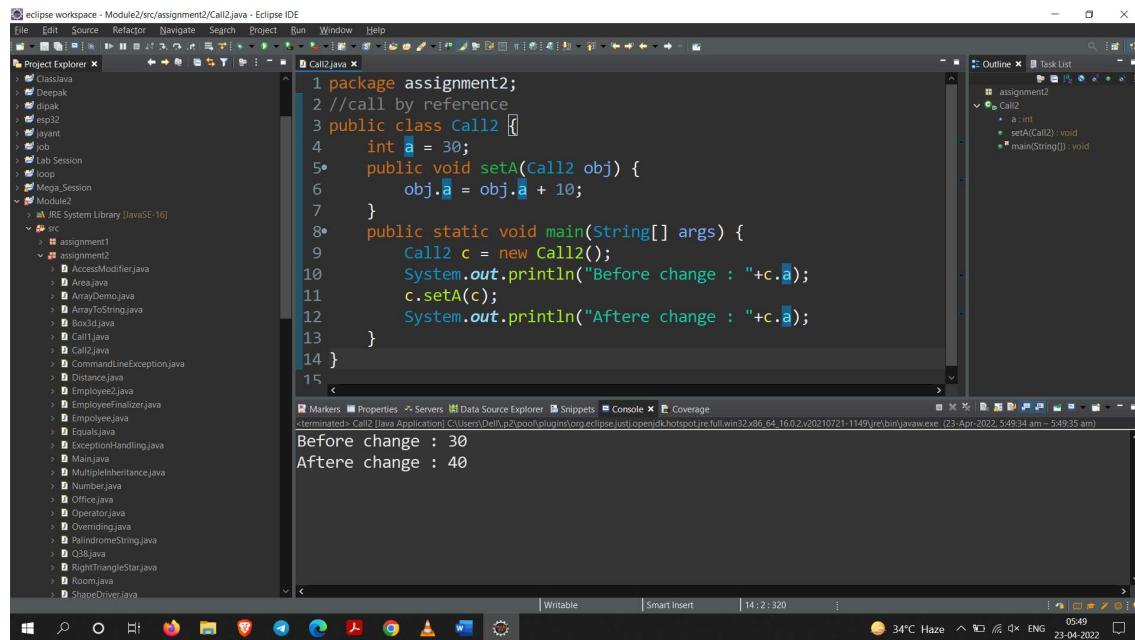
**Part – A**



```
1 package assignment2;
2 //Call by value
3 public class Call1 {
4     int a = 30;
5     public void setA(int a) {
6         a = a + 30;
7     }
8     public static void main(String[] args) {
9         Call1 c = new Call1();
10        System.out.println("Before Change : "+c.a);
11        c.setA(20);
12        System.out.println("After Change : "+c.a);
13    }
14 }
```

Before Change : 30  
After Change : 30

**Part – B**



```
1 package assignment2;
2 //call by reference
3 public class Call2 {
4     int a = 30;
5     public void setA(Call2 obj) {
6         obj.a = obj.a + 10;
7     }
8     public static void main(String[] args) {
9         Call2 c = new Call2();
10        System.out.println("Before change : "+c.a);
11        c.setA(c);
12        System.out.println("After change : "+c.a);
13    }
14 }
```

Before change : 30  
After change : 40

---

**Q31. Write a program to demonstrate the use of try, catch, finally throw and throws keywords and demonstrate the following points in the program.**

- a) Multiple catch blocks.
- b) try-catch-finally combination.
- c) try-finally combination.
- d) Exception propagation among many methods.
- e) Use of getMessage(), printStackTrace() function of
- 1. Throwable class.
- f) Nested try blocks

**Code:**

```
package assignment2;

import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

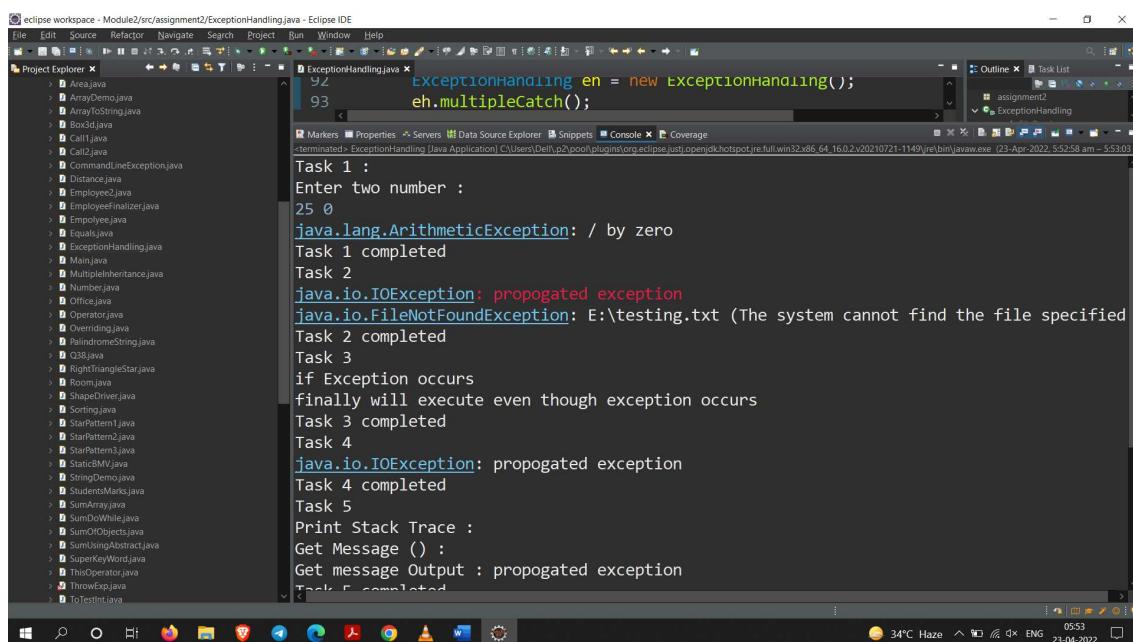
public class ExceptionHandling {
    FileReader f;
    Scanner sc = new Scanner(System.in);
    public void multipleCatch() {
        System.out.println("Task 1 : ");
        System.out.println("Enter two number : ");
        try {
            int x = Integer.parseInt(sc.next());
            int y = Integer.parseInt(sc.next());
            int z = x / y;
            System.out.println("Z : "+z);
        }catch(NumberFormatException n) {
            System.out.println(n);
        }catch(ArithmaticException e) {
            System.out.println(e);
        }
    }
    public void tryCatchFinally() throws IOException {
        System.out.println("Task 2");
        try{
            f = new FileReader("E://testing.txt");
            System.out.println(f+" is opened");
        }catch(Exception e) {
            System.out.println(e);
        }finally{
            if(f!=null) {
                f.close();
            }
        }
    }
}
```

```
public void tryFinally() {
    System.out.println("Task 3");
    try {
        System.out.println("if Exception occurs ");
    }finally {
        System.out.println("finally will execute even though exception occurs");
    }
}
public void n() throws IOException{
    throw new IOException("propogated exception");
}
public void a() throws IOException{
    n(); // propogated to n
}
public void ExceptionPropogation() {
    System.out.println("Task 4");
    try {
        a(); // propogated to a
    }catch(Exception e) {
        System.out.println(e);
    }
}
public void getMsg() {
    System.out.println("Task 5");
    try {
        n();
    } catch (IOException e) {
        System.out.println("Print Stack Trace : ");
        e.printStackTrace();
        System.out.println("Get Message () : ");
        System.out.println("Get message Output : "+e.getMessage());
    }
}
public void NestedTry() {
    System.out.println("Task 6");
    try {
        try {
            int a[] = new int[5];
            a[5] = 10;
        }catch(Exception e) {
            System.out.println(e);
        }
        try {
            int b = 10 / 0;
            System.out.println(b);
        }catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

```
        }catch(Exception e) {
            System.out.println(e);
        }
        System.out.println("Normal flow");
    }

    public static void main(String[] args) {
        ExceptionHandling eh = new ExceptionHandling();
        eh.multipleCatch();
        System.out.println("Task 1 completed");
        try {
            eh.tryCatchFinally();
        } catch(Exception e) {
            e.printStackTrace();
        }
        System.out.println("Task 2 completed");
        eh.tryFinally();
        System.out.println("Task 3 completed");
        eh.ExceptionPropogation();
        System.out.println("Task 4 completed");
        eh.getMsg();
        System.out.println("Task 5 completed");
        eh.NestedTry();
        System.out.println("Task 6 completed");
    }
}
```

### Output:



The screenshot shows the Eclipse IDE interface with the code editor displaying the `ExceptionHandling.java` file. The output window shows the execution of the program, which prints various messages and handles exceptions. The terminal output is as follows:

```
Task 1 :
Enter two number :
25 0
java.lang.ArithmetricException: / by zero
Task 1 completed
Task 2
java.io.IOException: propogated exception
java.io.FileNotFoundException: E:\testing.txt (The system cannot find the file specified)
Task 2 completed
Task 3
if Exception occurs
finally will execute even though exception occurs
Task 3 completed
Task 4
java.io.IOException: propogated exception
Task 4 completed
Task 5
Print Stack Trace :
Get Message () :
Get message Output : propogated exception
Task 5 completed
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a list of Java files in the assignment2 package, including Area.java, ArrayToString.java, Box3d.java, Call.java, CommandLineException.java, Distance.java, Employee2.java, EmployeeFinalizer.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern.java, StarPattern2.java, StarPattern3.java, StaticBMMV.java, StringDemo.java, StudentsMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, and ToTestmt.java.
- Code Editor:** Displays the ExceptionHandling.java file with the following code:

```
exceptionhandling en = new exceptionhandling();
eh.multipleCatch();

if Exception occurs
finally will execute even though exception occurs
Task 3 completed
Task 4
java.io.IOException: propogated exception
Task 4 completed
Task 5
Print Stack Trace :
Get Message () :
Get message Output : propogated exception
Task 5 completed
Task 6
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
java.lang.ArithmetricException: / by zero
Normal flow
Task 6 completed
```
- Outline View:** Shows the class structure: assignment2 > ExceptionHandling.
- Task List:** Lists tasks such as "Task 3 completed", "Task 4 completed", "Task 5 completed", and "Task 6 completed".
- Bottom Status Bar:** Shows system information: 34°C Haze, 0553, ENG, 23-04-2022.

**Q32. Write a program to throw a checked exception explicitly using 'throw' keyword and**

- a) Handle the exception in same method.
- b) use throws clause and handle the exception in some other method (calling method)
- c) Don't either handle or use the throws clause. Observe the result.

**Code:**

```
package assignment2;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

class Parrot {
    private Parrot(){}
}

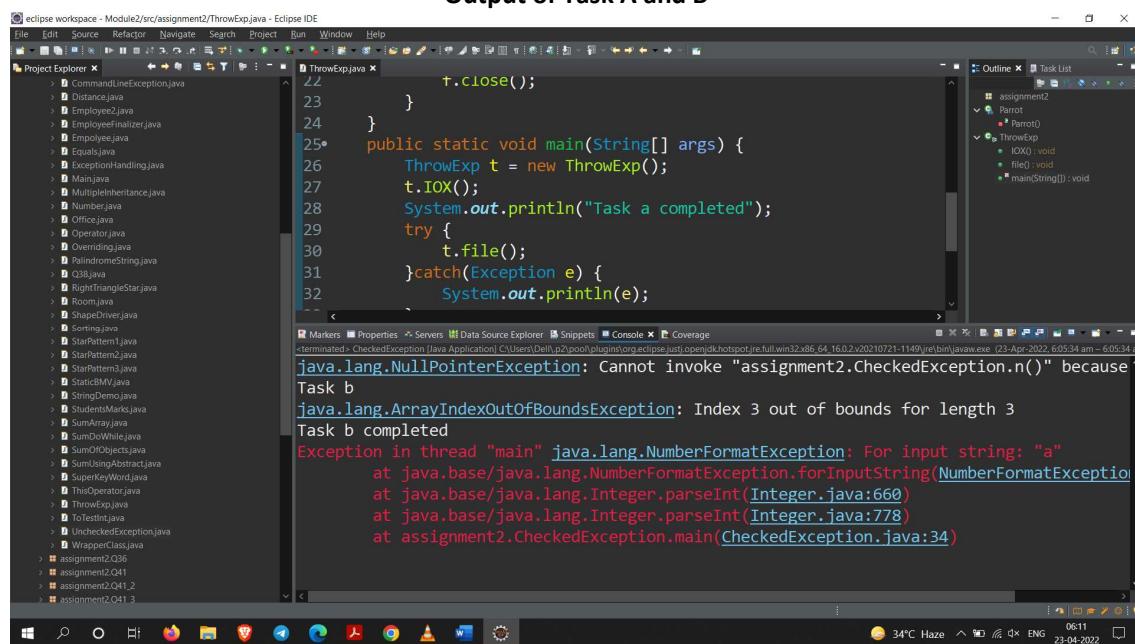
public class ThrowExp {
    public void IOX() {
        System.out.println("Task a");
        try {
            throw new IOException();
        }catch(IOException e) {
            System.out.println(e);
        }
    }

    public void file() throws FileNotFoundException,IOException{
        FileReader f = new FileReader("E:\\testing");
        if(f!=null) {
            f.close();
        }
    }

    public static void main(String[] args) {
        ThrowExp t = new ThrowExp();
        t.IOX();
        System.out.println("Task a completed");
        try {
            t.file();
        }catch(Exception e) {
            System.out.println(e);
        }
        System.out.println("Task b completed");
        System.out.println("Task c");
        Parrot p = new Parrot();
        System.out.println("Task c completed");
        System.out.println("Normal flow");
    }
}
```

### Output:

#### Output of Task A and B



```

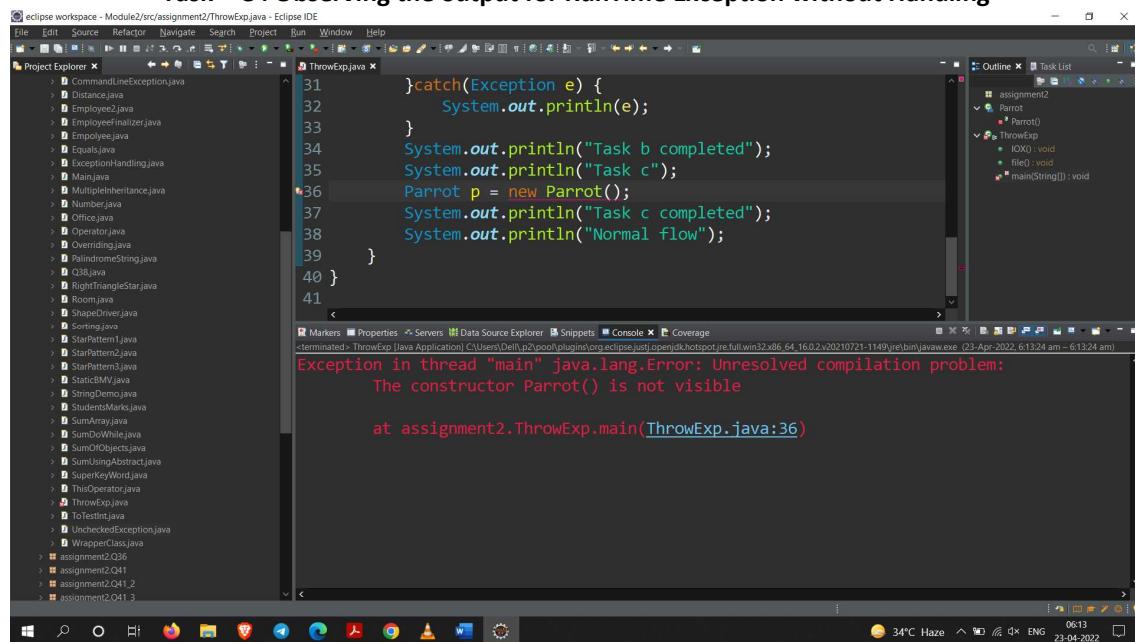
22     t.close();
23   }
24 }
25 public static void main(String[] args) {
26   ThrowExp t = new ThrowExp();
27   t.IOX();
28   System.out.println("Task a completed");
29   try {
30     t.file();
31   }catch(Exception e) {
32     System.out.println(e);
33   }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "assignment2.CheckedException.n()" because ^

Task b
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
Task b completed
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:66)
at java.base/java.lang.Integer.parseInt(Integer.java:778)
at java.base/java.lang.Integer.parseInt(Integer.java:778)
at assignment2.CheckedException.main(CheckedException.java:34)

```

#### Task – C : Observing the output for RunTime Exception without Handling



```

31   }catch(Exception e) {
32     System.out.println(e);
33   }
34   System.out.println("Task b completed");
35   System.out.println("Task c");
36   Parrot p = new Parrot();
37   System.out.println("Task c completed");
38   System.out.println("Normal flow");
39 }
40 }
41 }

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
  The constructor Parrot() is not visible
  at assignment2.ThrowExp.main(ThrowExp.java:36)

```

**Q33. Repeat program 32 with unchecked Exception and demonstrate the difference in both program.**

**Code:**

```
package assignment2;

public class UncheckedException {
    public void n() {
        System.out.println("n");
    }
    @SuppressWarnings("null")
    public static void nullPointer(){
        UncheckedException uc = new UncheckedException();
        uc = null;
        try{
            uc.n();
        }catch(Exception e) {
            System.out.println(e);
        }
    }
    public void arrayindex(int a[]) throws Exception{
        System.out.println("Task b");
        a[0] = 1;
        a[1] = 2;
        a[2] = 3;
        a[3] = 4;
    }
    public static void main(String[] args) {
        nullPointer();
        int a[] = new int[3];
        UncheckedException u = new UncheckedException();
        try{
            u.arrayindex(a);
        }catch(Exception e) {
            System.out.println(e);
        }
        System.out.println("Task b completed");
        int i = Integer.parseInt("a");
        System.out.println(i);
        System.out.println("Task c completed");
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a large list of Java files under the package `assignment2`, including `Number.java`, `Office.java`, `Operator.java`, `Overriding.java`, `PallindromeString.java`, `Q3.java`, `RightTriangleStar.java`, `Room.java`, `ShapeDriver.java`, `Sorting.java`, `StarPattern1.java`, `StarPattern2.java`, `StarPattern3.java`, `StaticDMV.java`, `StringDemo.java`, `StudentsMarks.java`, `SumArray.java`, `SumDoWhile.java`, `SumOfObjects.java`, `SumUsingAbstract.java`, `SuperKeyWord.java`, `ThisOperator.java`, `ThrowExp.java`, `ToTest.java`, `UncheckedException.java`, and `WrapperClass.java`.
- UnCheckedException.java:** The code in this file is as follows:

```
System.out.println("Task b completed");
int i = Integer.parseInt("a");
System.out.println(i);
System.out.println("Task c completed");
```
- Console Output:** The output window shows the following errors:

```
java.lang.NullPointerException: Cannot invoke "assignment2.UncheckedException.n()" because the return type of this expression is void
Task b
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
Task b completed
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:66)
at java.base/java.lang.Integer.parseInt(Integer.java:660)
at java.base/java.lang.Integer.parseInt(Integer.java:778)
at assignment2.UncheckedException.main(UncheckedException.java:34)
```
- System Tray:** Shows icons for battery (34°C Haze), network, volume, and system.

**Q34. Write a program to give example for multiple inheritance in Java.**

**Code:**

```
package assignment2;

interface I1{
    void print();
}
interface I2{
    void show();
}
class X implements I1,I2{ // multiple inheritance with interfaces
    public void print(){
        System.out.println("A");
    }

    public void show(){
        System.out.println("B");
    }
}
public class MultipleInheritance{
    public static void main(String[] args) {
        X m=new X();
        m.print();
        m.show();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Assignment2, I1, I2, X, and MultipleInheritance.
- Code Editor:** Displays the source code for `MultipleInheritance.java`.
- Outline View:** Shows the class structure with methods `print()` and `show()` for class `X`.
- Output View:** Shows the console output with the text "A" and "B" printed sequentially.
- System Tray:** Shows system information like temperature (34°C), battery level, and network status.

**Q35. Write a program to demonstrate the use of equals method of Object class and compare its functionality with ( == ) operator.**

**Code:**

```
package assignment2;

public class Equals {
    public static void main(String[] args)
    {
        String s1 = "HELLO";
        String s2 = "HELLO";
        String s3 = new String("HELLO");

        System.out.println(s1 == s2);
        System.out.println(s1 == s3);
        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Assignment2.java, Equals.java, and many other files related to assignments and operators.
- Code Editor:** Displays the `Equals.java` source code.
- Console:** Shows the output of the program execution:

```
true
false
true
true
```

**Q36. Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.**

**Code:**

```
package assignment2.Q36;
class A{
    public void test() {
        System.out.println("Method 1");
    }
}
class B{
    public void test2() {
        System.out.println("Method 2");
    }
}
class C{
    public void test3() {
        System.out.println("Method 3");
    }
}
public class Reference {
    public static void main(String[] args) {
        A a1 = new A();
        B b1 = new B();
        C c1 = new C();
        a1.test();
        b1.test2();
        c1.test3();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Assignment2.Q36, Main.java, MultipInheritance.java, Number.java, Operator.java, Overriding.java, PalindromeString.java, Q36.java, RightTriangleStar.java, Room.java, ShapedDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, WrapperClass.java, and several assignment-related files.
- Reference.java:** The code block above is pasted into this file.
- Outline View:** Shows the class structure:
  - assignment2.Q36
    - A
      - test():void
    - B
      - test2():void
    - C
      - test3():void
    - Reference
      - main(String[]):void
- Console:** Displays the output of the program:

```
Method 1
Method 2
Method 3
```
- System Tray:** Shows the date and time as 23-Apr-2022, 6:19:28 am.

**Q37. Create String Demo class and perform different string manipulation methods.**

**Code:**

```
package assignment2;

public class StringDemo {
    public static void main(String[] args) {
        String str = "Sarang";
        System.out.println(str.toLowerCase());
        System.out.println(str.toUpperCase());
        System.out.println(str.replace('a', 'w'));
        System.out.println(str.charAt(3));
        System.out.println(str.length());
        System.out.println(str.contains("a"));
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Assignment2, Main.java, MultipelInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q38.java, RightAngleStar.java, Rotate.java, ShapesFigure.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, WrapperClass.java, and several sub-packages under assignment2.
- Code Editor:** Displays the StringDemo.java code.
- Console:** Shows the output of the program:

```
sarang
SARANG
Swrwng
a
6
true
```
- Bottom Status Bar:** Shows system information: 34°C Haze, 06:21, ENG, 23-04-2022.

**Q38. Create sample classes to understand boxing & unboxing.**

**Code:**

```
package assignment2;
class AutoBoxing{
    public void boxing(){
        Character a = 'a';
        char c = a;
        System.out.println(c);
    }
}
class UnBoxing{
    @SuppressWarnings("removal")
    public void unBoxing() {
        Integer i = new Integer(10);
        int i1 = i.intValue();
        System.out.println(i1);
    }
}
public class Q38 {
    public static void main(String[] args) {
        AutoBoxing a = new AutoBoxing();
        UnBoxing u = new UnBoxing();
        a.boxing();
        u.unBoxing();
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like Lab Session, loop, Mega\_Session, Module2, and assignment2.
- Code Editor:** Displays the Java code for Q38.java, which contains the classes AutoBoxing and UnBoxing.
- Outline View:** Shows the class hierarchy: assignment2, AutoBoxing, UnBoxing, and Q38.
- Console View:** Shows the output of the program:

```
a
10
```
- System Tray:** Shows system status including temperature (34°C), battery level, and network connection.

**Q39. Use different methods of java defined wrapper classes.**

**Code:**

```
package assignment2;

public class WrapperClass {
    public static void main(String[] args) {
        Integer i1 = Integer.parseInt("10");
        System.out.println(i1);
        Float f1 = Float.valueOf("20");
        System.out.println(f1);
        Integer i2 = Integer.valueOf(20);
        System.out.println(i2);
        String i3 = Long.toString(10,2);
        System.out.println(i3);
    }
}
```

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files in the workspace, including Employee.java, Equals.java, ExceptionHandling.java, Main.java, MultipleInheritance.java, Number.java, Office.java, Operator.java, Overriding.java, PalindromeString.java, Q3.java, RightTriangleStar.java, Room.java, ShapeDriver.java, Sorting.java, StarPattern1.java, StarPattern2.java, StarPattern3.java, StaticCMV.java, StringDemo.java, StudentMarks.java, SumArray.java, SumDoWhile.java, SumOfObjects.java, SumUsingAbstract.java, SuperKeyWord.java, ThisOperator.java, ThrowExp.java, ToTest.java, UncheckedException.java, and WrapperClass.java.
- Editor:** Displays the code for `WrapperClass.java`. The code uses `Integer.parseInt()`, `Float.valueOf()`, and `Long.toString()` methods to convert strings to their respective wrapper class objects and then prints them.
- Outline View:** Shows the class structure with the `main(String[] args)` method highlighted.
- Console:** Shows the terminal output of the program execution:  
10  
20.0  
20  
1010
- System Tray:** Shows system status including temperature (34°C), battery level, and network connection.

---

**Q40. Create a class Employee and encapsulate the data members.**

**Code:**

```
package assignment2;
class Edata {
    String name;
    int id;
    float salary;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public float getSalary() {
        return salary;
    }
    public void setSalary(float salary) {
        this.salary = salary;
    }
}
public class Employee2 {
    public static void main(String[] args) {
        Edata e = new Edata();
        e.setName("Sarang");
        e.setId(10);
        e.setSalary(100000.0f);
        System.out.println("Name : "+e.getName());
        System.out.println("Id No. : "+e.getId());
        System.out.println("Salary : "+e.getSalary());
    }
}
```

Roll No.: 220350320026  
Name: Sarang G. Deodhar  
PG-DAC March 2022  
Module 2: Object Oriented Programming in JAVA

**Output:**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under Module2, including the src folder which contains assignment1, assignment2, and Employee2.java.
- Employee2.java Content:** The code defines a class Employee2 with a main method. It creates an object e of type Edata, sets its name to "Sarang", id to 10, and salary to 100000.0f, then prints these values.
- Outline View:** Shows the class Edata with its attributes (name, id, salary) and methods (setName, setId, setSalary, getName, getId, getSalary).
- Console Output:** Displays the printed values:  
Name : Sarang  
Id No. : 10  
Salary : 100000.0
- System Tray:** Shows the date (23-Apr-2022), time (06:27), and weather (34°C Haze).

---

**Q41. Implement multilevel inheritance using different packages.**

**Code:**

**Part – A**

```
package assignment2.Q41;

public class Base {
    public int i = 10;
    public void test() {
        System.out.println("Test");
    }
}
```

**Part – B**

```
package assignment2.Q41_2;

import assignment2.Q41.Base;

public class Child extends Base {
    public int z = 20;
    public void test2() {
        System.out.println("From Child-Base (intermediate)");
    }
    public static void main(String[] args) {
        Child c = new Child();
        c.test();
        System.out.println(c.i);
    }
}
```

**Part – C**

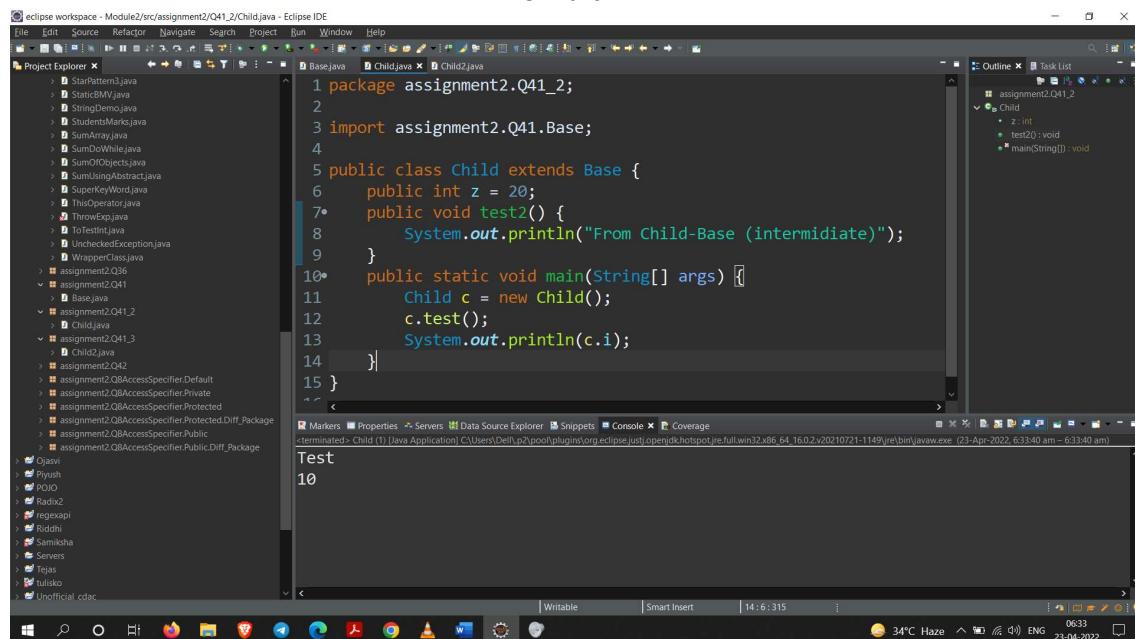
```
package assignment2.Q41_2;

import assignment2.Q41.Base;

public class Child extends Base {
    public int z = 20;
    public void test2() {
        System.out.println("From Child-Base (intermediate)");
    }
    public static void main(String[] args) {
        Child c = new Child();
        c.test();
        System.out.println(c.i);
    }
}
```

**Output:**

**For Part – B**

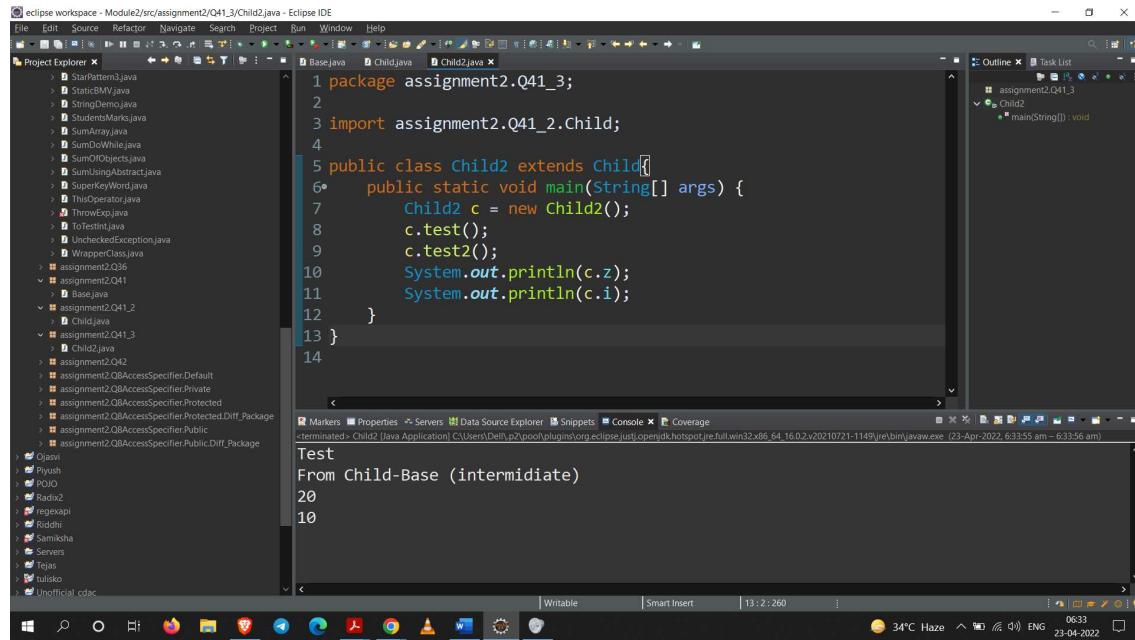


```

1 package assignment2.Q41_2;
2
3 import assignment2.Q41.Base;
4
5 public class Child extends Base {
6     public int z = 20;
7     public void test2() {
8         System.out.println("From Child-Base (intermediate)");
9     }
10    public static void main(String[] args) {
11        Child c = new Child();
12        c.test();
13        System.out.println(c.i);
14    }
15 }

```

**For Part – C**



```

1 package assignment2.Q41_3;
2
3 import assignment2.Q41_2.Child;
4
5 public class Child2 extends Child{
6     public static void main(String[] args) {
7         Child2 c = new Child2();
8         c.test();
9         c.test2();
10        System.out.println(c.z);
11        System.out.println(c.i);
12    }
13 }

```

**Q42. Access/invoke protected members/methods of a class outside the package.**

**Code:**

**Part – A**

```
package assignment2.Q42;

public class Base1 {
    protected int member = 20;
    public void method() {
        System.out.println("From Parent class");
    }
}
```

**Part – B**

```
package assignment2.Q42_2;

import assignment2.Q42.Base1;

public class Child3 extends Base1{
    public static void main(String[] args) {
        Child1 c = new Child1();
        c.method();
        System.out.println(c.member);
    }
}
```

**Output:**

```
1 package assignment2.Q42_2;
2
3 import assignment2.Q42.Base1;
4
5 public class Child1 extends Base1{
6     public static void main(String[] args) {
7         Child1 c = new Child1();
8         c.method();
9         System.out.println(c.member);
10    }
11 }
```

From Parent class  
20