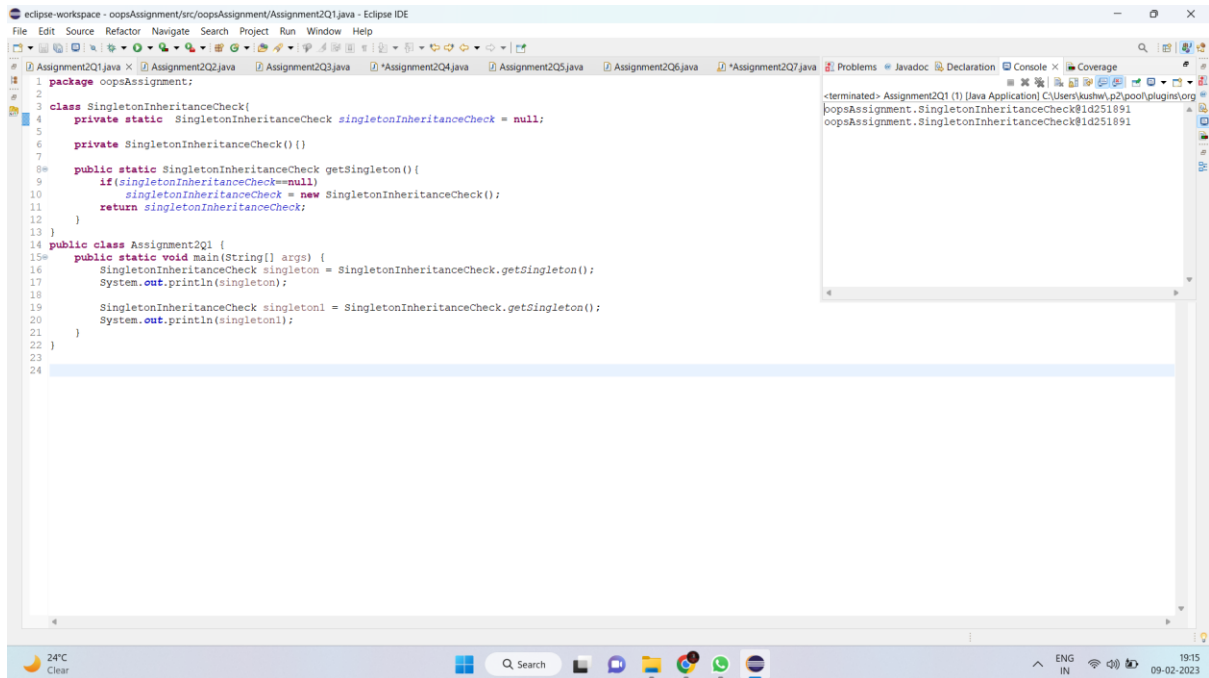


Intermediate OOPS Assignment

Assignment2Q1



The screenshot shows the Eclipse IDE with the file `Assignment2Q1.java` open. The code implements a Singleton pattern for `SingletonInheritanceCheck` and a `main` method in `Assignment2Q1` that demonstrates its use. The console on the right shows the output of the program, indicating that the singleton instance was successfully retrieved and printed.

```
package oopsAssignment;

class SingletonInheritanceCheck {
    private static SingletonInheritanceCheck singletonInheritanceCheck = null;

    private SingletonInheritanceCheck() {}

    public static SingletonInheritanceCheck getSingleton() {
        if (singletonInheritanceCheck == null) {
            singletonInheritanceCheck = new SingletonInheritanceCheck();
        }
        return singletonInheritanceCheck;
    }
}

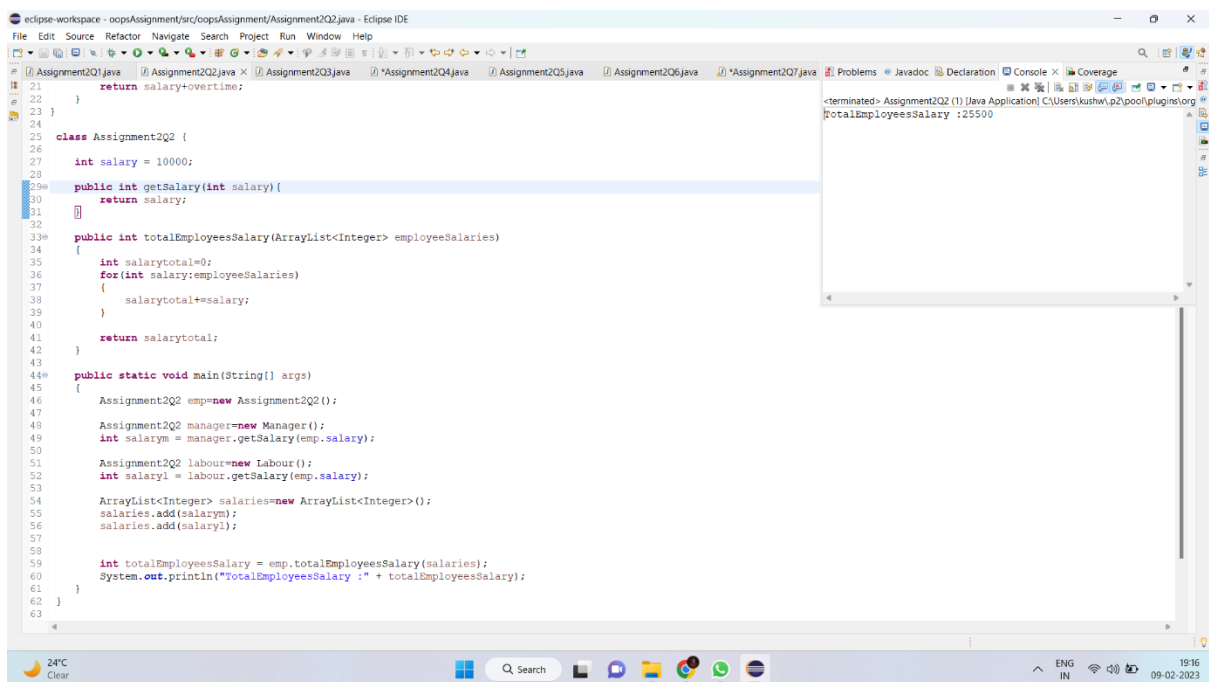
public class Assignment2Q1 {
    public static void main(String[] args) {
        SingletonInheritanceCheck singleton = SingletonInheritanceCheck.getSingleton();
        System.out.println(singleton);

        SingletonInheritanceCheck singleton1 = SingletonInheritanceCheck.getSingleton();
        System.out.println(singleton1);
    }
}
```

Console Output:

```
<terminated> Assignment2Q1 (1) [Java Application] C:\Users\kushw.p2\pool\plugins\org
oopsAssignment.SingletonInheritanceCheck#1d251891
oopsAssignment.SingletonInheritanceCheck#1d251891
```

Assignment2Q2



The screenshot shows the Eclipse IDE with the file `Assignment2Q2.java` open. The code defines a `Manager` class with a `getSalary` method and a `totalEmployeesSalary` method that calculates the total salary for a list of employees. The `main` method demonstrates the usage of these classes. The console on the right shows the output of the program, displaying the total employees salary as 25500.

```
return salary+overtime;
}

class Assignment2Q2 {
    int salary = 10000;

    public int getSalary(int salary) {
        return salary;
    }

    public int totalEmployeesSalary(ArrayList<Integer> employeeSalaries) {
        int salarytotal=0;
        for(int salary:employeeSalaries) {
            salarytotal+=salary;
        }
        return salarytotal;
    }

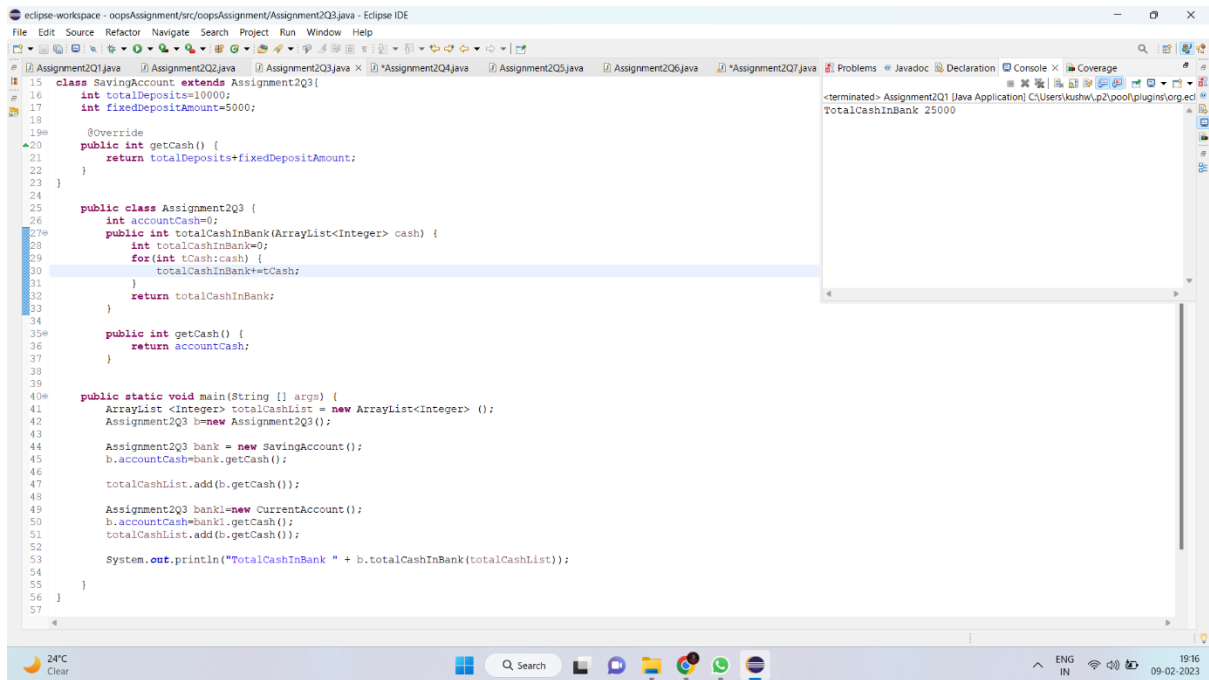
    public static void main(String[] args) {
        Assignment2Q2 emp=new Assignment2Q2();
        Assignment2Q2 manager=new Manager();
        int salarym = manager.getSalary(emp.salary);
        Assignment2Q2 labour=new Labour();
        int salaryl = labour.getSalary(emp.salary);
        ArrayList<Integer> salaries=new ArrayList<Integer>();
        salaries.add(salarym);
        salaries.add(salaryl);

        int totalEmployeesSalary = emp.totalEmployeesSalary(salaries);
        System.out.println("TotalEmployeesSalary :"+ totalEmployeesSalary);
    }
}
```

Console Output:

```
<terminated> Assignment2Q2 (1) [Java Application] C:\Users\kushw.p2\pool\plugins\org
TotalEmployeesSalary :25500
```

Assignment2Q3

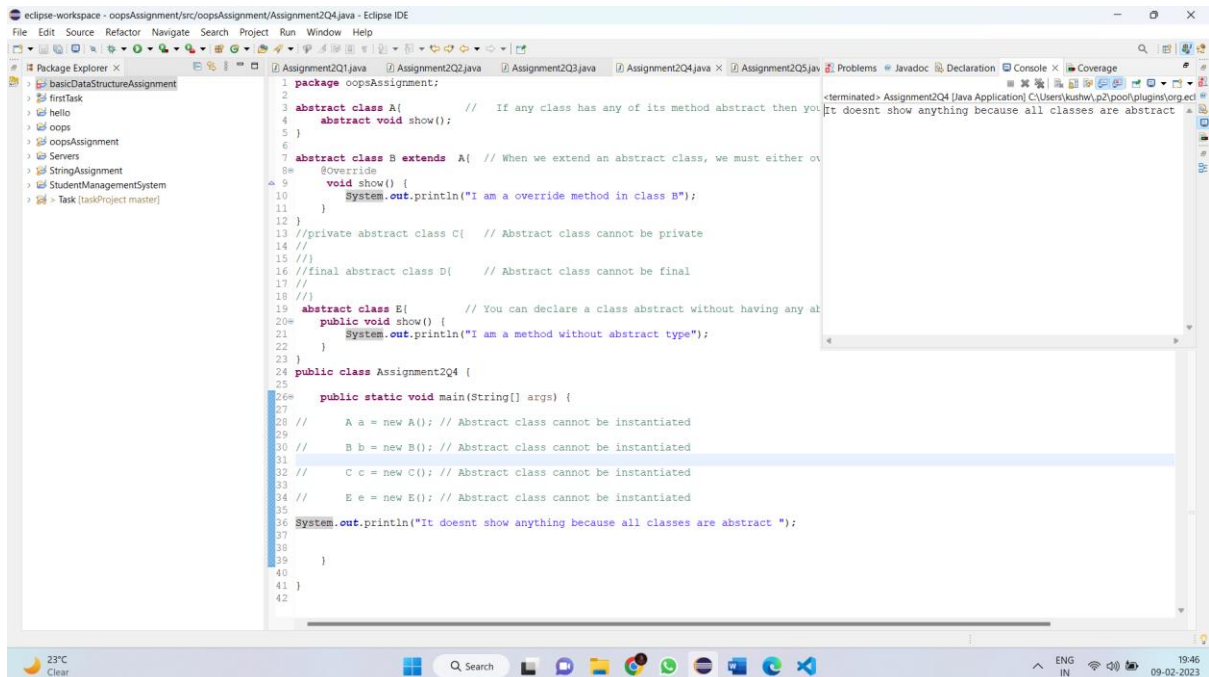


The screenshot shows the Eclipse IDE with the file `Assignment2Q3.java` open. The code defines a `SavingAccount` class that extends `Assignment2Q3`, and an `Assignment2Q3` class with a `totalCashInBank` method. The `main` method creates a `SavingAccount` object and a `CurrentAccount` object, and prints the total cash in bank.

```
15 class SavingAccount extends Assignment2Q3 {
16     int totalDeposits=10000;
17     int fixedDepositAmount=5000;
18
19     @Override
20     public int getCash() {
21         return totalDeposits+fixedDepositAmount;
22     }
23 }
24
25 public class Assignment2Q3 {
26     int accountCash=0;
27     public int totalCashInBank(ArrayList<Integer> cash) {
28         int totalCashInBank=0;
29         for(int tCash:cash) {
30             totalCashInBank+=tCash;
31         }
32         return totalCashInBank;
33     }
34
35     public int getCash() {
36         return accountCash;
37     }
38
39
40     public static void main(String [] args) {
41         ArrayList<Integer> totalCashList = new ArrayList<Integer> ();
42         Assignment2Q3 b=new Assignment2Q3();
43
44         Assignment2Q3 bank = new SavingAccount();
45         b.accountCash=bank.getCash();
46
47         totalCashList.add(b.getCash());
48
49         Assignment2Q3 bank1=new CurrentAccount();
50         b.accountCash=bank1.getCash();
51         totalCashList.add(b.getCash());
52
53         System.out.println("TotalCashInBank " + b.totalCashInBank(totalCashList));
54     }
55 }
56
57
```

The console output shows: `<terminated>- Assignment2Q1 [Java Application] C:\Users\kushwlp2\pool\plugins\org.ej TotalCashInBank 25000`

Assignment2Q4

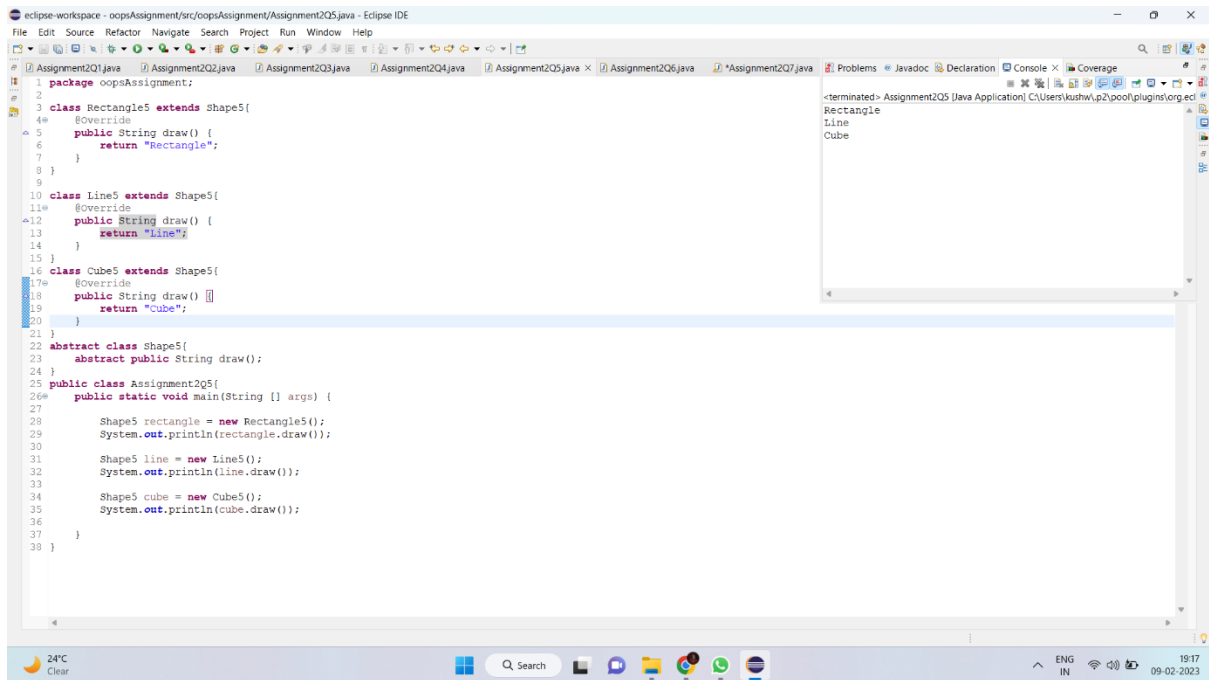


The screenshot shows the Eclipse IDE with the file `Assignment2Q4.java` open. The code defines an abstract class `A` with an abstract method `show()`, and several concrete classes `B`, `C`, `D`, and `E` that extend `A`. The `main` method creates objects of these classes and prints the output of the `show()` method.

```
1 package oopsAssignment;
2
3 abstract class A{ // If any class has any of its method abstract then you
4     abstract void show();
5 }
6
7 abstract class B extends A{ // When we extend an abstract class, we must either ov
8     @Override
9     void show() {
10         System.out.println("I am a override method in class B");
11     }
12 }
13 //private abstract class C{ // Abstract class cannot be private
14 //
15 //
16 //final abstract class D{ // Abstract class cannot be final
17 //
18 //
19 abstract class E{ // You can declare a class abstract without having any ab
20     public void show() {
21         System.out.println("I am a method without abstract type");
22     }
23 }
24 public class Assignment2Q4 {
25
26     public static void main(String[] args) {
27
28         // A a = new A(); // Abstract class cannot be instantiated
29
30         // B b = new B(); // Abstract class cannot be instantiated
31
32         // C c = new C(); // Abstract class cannot be instantiated
33
34         // E e = new E(); // Abstract class cannot be instantiated
35
36         System.out.println("It doesnt show anything because all classes are abstract ");
37
38     }
39 }
40
41 }
42
```

The console output shows: `<terminated>- Assignment2Q4 [Java Application] C:\Users\kushwlp2\pool\plugins\org.ej It doesnt show anything because all classes are abstract`

Assignment2Q5



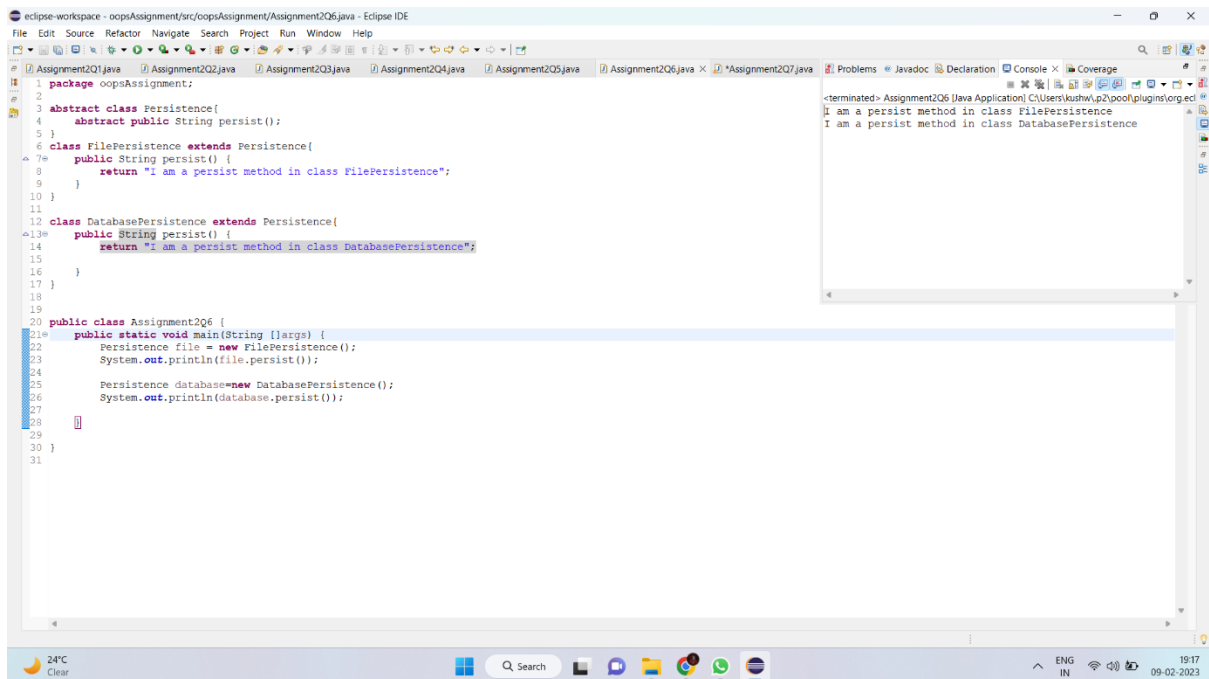
The screenshot shows the Eclipse IDE with the file `Assignment2Q5.java` open. The code defines an abstract class `Shape5` with an abstract method `draw()`. Two subclasses, `Rectangle5` and `Line5`, override this method to return "Rectangle" and "Line" respectively. A third subclass, `Cube5`, overrides it to return "Cube". The `main` method in `Assignment2Q5` creates instances of these classes and prints their `draw()` results.

```
1 package oopsAssignment;
2
3 class Rectangle5 extends Shape5{
4     @Override
5     public String draw() {
6         return "Rectangle";
7     }
8 }
9
10 class Line5 extends Shape5{
11     @Override
12     public String draw() {
13         return "Line";
14     }
15 }
16 class Cube5 extends Shape5{
17     @Override
18     public String draw() {
19         return "Cube";
20     }
21 }
22 abstract class Shape5{
23     abstract public String draw();
24 }
25 public class Assignment2Q5{
26     public static void main(String [] args) {
27
28         Shape5 rectangle = new Rectangle5();
29         System.out.println(rectangle.draw());
30
31         Shape5 line = new Line5();
32         System.out.println(line.draw());
33
34         Shape5 cube = new Cube5();
35         System.out.println(cube.draw());
36
37     }
38 }
```

The right-hand pane shows the output of the program:

```
<terminated>- Assignment2Q5 [Java Application] C:\Users\kushwlp2\pool\plugins\org.ecl
Rectangle
Line
Cube
```

Assignment2Q6



The screenshot shows the Eclipse IDE with the file `Assignment2Q6.java` open. The code defines an abstract class `Persistence` with an abstract method `persist()`. Two subclasses, `FilePersistence` and `DatabasePersistence`, override this method to return "I am a persist method in class FilePersistence" and "I am a persist method in class DatabasePersistence" respectively. The `main` method in `Assignment2Q6` creates instances of these classes and prints their `persist()` results.

```
1 package oopsAssignment;
2
3 abstract class Persistence{
4     abstract public String persist();
5 }
6 class FilePersistence extends Persistence{
7     public String persist() {
8         return "I am a persist method in class FilePersistence";
9     }
10 }
11
12 class DatabasePersistence extends Persistence{
13     public String persist() {
14         return "I am a persist method in class DatabasePersistence";
15     }
16 }
17
18
19
20 public class Assignment2Q6 {
21     public static void main(String []args) {
22         Persistence file = new FilePersistence();
23         System.out.println(file.persist());
24
25         Persistence database=new DatabasePersistence();
26         System.out.println(database.persist());
27
28     }
29 }
30 }
31 }
```

The right-hand pane shows the output of the program:

```
<terminated>- Assignment2Q6 [Java Application] C:\Users\kushwlp2\pool\plugins\org.ecl
I am a persist method in class FilePersistence
I am a persist method in class DatabasePersistence
```

Assignment2Q7

The screenshot displays the Eclipse IDE interface. The main editor window shows the source code for `Assignment2Q7.java`. The code implements a program where a user enters their role (customer or owner) and then selects an item (candy, cookie, or ice) to purchase. It calculates the total cost based on predefined prices and currency conversion rates.

```
36 System.out.println("Enter the how much Ice do you want to add");
37 int ice=ssc.nextInt();
38 System.out.println("Ice add " + ice);
39 return 0;
40 }
41
42 public static void main(String [] args) {
43     Assignment2Q7 Al=new Assignment2Q7();
44     Scanner sc = new Scanner(System.in);
45     System.out.println("Enter your role customer or owner");
46     String s = sc.nextLine();
47     System.out.println("Select 1 for seller and Select 2 for Buyer");
48     int num=ssc.nextInt();
49
50     if (num==1) {
51         Al.addCandy();
52         Al.addCookie();
53         Al.addIce();
54     }
55     else if (num==2) {
56         System.out.println("Enter how much item you want to purchase");
57         int n=ssc.nextInt();
58         System.out.println("Enter how much candy buy ");
59         int candy = sc.nextInt();
60         System.out.println("Enter how much cookie buy ");
61         int cookie = sc.nextInt();
62         System.out.println("Enter how much ice buy ");
63         int ice = sc.nextInt();
64         int a1=10;
65         int a2=20;
66         int a3=30;
67         System.out.println("The cost of 1 candy : " + a1);
68
69         System.out.println("The cost of 1 cookie : "+a2);
70
71         System.out.println("The cost of 1 ice : "+a3);
72
73         int s1= 60 * (a1*candy); // 1 dollar = 60 Rupees
74         int s2= 70 * (a2*cookie); // 1 euro = 70 Rupees
75         int s3= a3*ice;
76         int total = s1+s2+s3;
77
78     }
```

The console window on the right shows the execution output:

```
<terminated> Assignment2Q7 (Java Application) C:\Users\kushw\p2\poo\plugins\org.ed
customer
Enter your role customer or owner
Select 1 for seller and Select 2 for Buyer
2
Enter how much item you want to purchase
10
Enter how much candy buy
5
Enter how much cookie buy
10
Enter how much ice buy
15
The cost of 1 candy : 10
The cost of 1 cookie : 20
The cost of 1 ice : 30
Total cost = 17450
```