NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Cachar, Assam

B.Tech. IVth Sem

Subject Code: CS212

Subject Name: Object Oriented Programming

Submitted By:

Name : Subhojit Ghimire

Sch. Id. : 1912160

Branch : CSE – B

1. **Write a program to count the number of vowels and consonants in a string. Make separate methods for counting.**

➔
```
import java.util.*;
import java.lang.*;

class CountVowelConsonant {
   public static int countVowel (int vowel, String input) {
      for (int i=0; i<input.length(); ++i) {
         if (input.charAt(i) == 'A' || input.charAt(i) == 'E' || input.charAt(i) == 'I' ||
input.charAt(i) == 'O' || input.charAt(i) == 'U')
            ++vowel;
      }
      return (vowel);
   }
   public static int countCons (int cons, String input) {
      for (int i=0; i<input.length(); ++i) {
         if (!(input.charAt(i) == 'A' || input.charAt(i) == 'E' || input.charAt(i) == 'I' ||
input.charAt(i) == 'O' || input.charAt(i) == 'U'))
            if (input.charAt(i) > 'A' && input.charAt(i) < 'Z')
            ++cons;
      }
      return (cons);
   }
   public static void main (String[] args) {
      Scanner Scan = new Scanner (System.in);
      System.out.print ("Enter String: ");
      String input = Scan.nextLine ();
      input = input.toUpperCase();
      int vowel = 0, cons = 0;
      System.out.println ("Number of Vowels in the string = " + countVowel (vowel,
input));
      System.out.println ("Number of Consonants in the string = " + countCons (cons,
input));
     Scan.close();
   }
}
```

2. Write a program to sort an array without using in-built function.

➔ import java.util.Scanner;

```
public class BubbleSort {
   public static void bubbleSort (int[] arr) {
      for (int i=0; i<arr.length - 1; ++i) {
         for (int j=0; j<arr.length - 1 - i; ++j) {
            if (arr[j] > arr[j+1]) {
               int temp = arr[j];
               arr[j] = arr[j+1];
               arr[j+1] = temp;
            }
         }
      }
   }
   public static void main (String[] args) {
      Scanner Scan = new Scanner (System.in);
      System.out.print ("Enter the size fo the array: ");
      int size = Scan.nextInt();
      int[] inputs = new int[size];
      for (int i=0; i<size; ++i) {
         System.out.print ("> ");
         inputs[i] = Scan.nextInt();
      }
      bubbleSort (inputs);
      System.out.print ("Sorted: ");
      for (int i=0; i<size; ++i)
         System.out.print (inputs[i] + " ");
      Scan.close();
   }
}
```

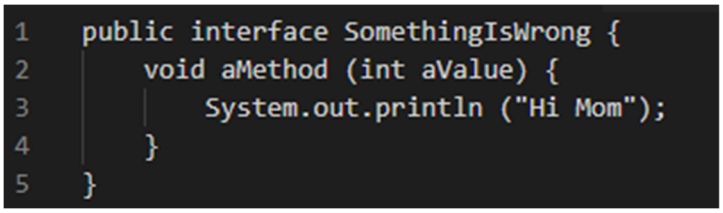
3. **Consider an outer class Area with two members: dimension1, dimension2. The outer class also has three inner classes: Circle, Rectangle and Triangle where the area of the corresponding shapes are computed by accessing the variables of the outer class.**

➔

```
import java.util.Scanner;

class Area {
  public static float dimension1, dimension2;

  private class Circle {
    public Circle () {
      System.out.println ("The area of the circle is: " + Math.PI*Math.pow(dimension1,
2));
    }
  }

  private class Rectangle {
    public Rectangle () {
      System.out.println ("The area of the rectangle is: " + dimension1*dimension2);
    }
  }

  private class Triangle {
    public Triangle () {
      System.out.println ("The area of the triangle is: " +
(0.5*dimension1*dimension2));
    }
  }

  void areaCircle () {
    Circle objCircle = new Circle();
  }

  void areaRectangle () {
    Rectangle objRect = new Rectangle();
  }

  void areaTriangle () {
    Triangle objTri = new Triangle();
  }
}

public class ComputeArea {
  public static void main (String[] args) {
    Scanner Scan = new Scanner(System.in);
    Area objArea = new Area();
    System.out.print ("Enter Dimension 1: ");
    objArea.dimension1 = Scan.nextInt();
```

```
        System.out.print ("Enter Dimenstion 2: ");
        objArea.dimension2 = Scan.nextInt();
        objArea.areaCircle();
        objArea.areaRectangle();
        objArea.areaTriangle();
       Scan.close();
    }
}
```

4. Fix the error in the interface:



```
1   public interface SomethingIsWrong {
2       void aMethod (int aValue) {
3           System.out.println ("Hi Mom");
4       }
5   }
```

➔ Fix 1: Removing method implementation:

```
public interface SomethingIsWrong {
   void aMethod (int aValue);
}
```

Fix 2: Defining aMethod as a default method (because, only default and static methods can have implementation):

```
public interface SomethingIsWrong {
   default void aMethod (int aValue) {
      System.out.println ("Hi Mom");
   }
}
```

5. **What will be the output of this program? Justify your answer:**

```
1   class A {
2       static {
3           System.out.println ("THIRD");
4       }
5   }
6
7   class B extends A {
8       static {
9           System.out.println ("SECOND");
10      }
11  }
12
13  class C extends B {
14      static {
15          system.out.println ("FIRST");
16      }
17  }
18
19  public class MainClass {
20      public static void main (String[] args) {
21          C c = new C();
22      }
23  }
```

➔ **OUTPUT:** THIRD

SECOND

FIRST

Explanation: In the main class, the object of class C is created, meaning it should call the constructor of class C first. But here, inheritance is used. C has inherited characteristics of class B, meaning, class B's constructor will be called when the object of C is created. But again, B is inherited from class A, meaning A is the parent class of B and B is the parent class of C. This means, when the object of C is created, the constructor of A is called first, then the constructor of B is called, and lastly the constructor of C is executed.

6. **What will be the output of this program? Justify your answer.**

```
1   class A {
2       int i = 10;
3   }
4
5   class B extends A {
6       int i = 20;
7       float j = 30;
8   }
9
10  public class MainClass {
11      public static void main (String[] args) {
12          A a = new B();
13          System.out.println (a.i);
14          System.out.println (a.j);
15      }
16  }
```

➔ Given 'a' is an object for class A, and 'j' is not declared in that class, System.out.println (a.j) will give an error message.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        j cannot be resolved or is not a field

        at MainClass.main(MainClass.java:14)
```

However, assuming the code in line 12 was:

B a = new B();

The output would be: 20
30.0

7. Consider the following:

```
1   public class ClassA {
2       public void methodOne (int i) {
3       }
4       public void methodTwo (int i) {
5       }
6       public static void methodThree (int i) {
7       }
8       public static void methodFour (int i) {
9       }
10  }
11
12  public class ClassB extends ClassA {
13      public static void methodOne (int i) {
14      }
15      public void methodTwo (int i) {
16      }
17      public void methodThree (int i) {
18      }
19      public static void methodFour (int i) {
20      }
21  }
```

a. Which method overrides a method in the superclass?

➔ methodTwo is capable of overriding the corresponding method in the superclass ClassA.
Reason: An instance method overrides superclass' instance method.

b. Which method hides a method in the superclass?

➔ methodFour is capable of hiding the corresponding method in the superclass ClassA.
Reason: A static method hides superclass' static method.

c. What do other methods do?

➔ methodOne and methodThree generate a compile-time error.

The results of this question can be summarised with this table:

	Superclass Instance Method	**Superclass Static Method**
Subclass Instance Method	Overrides	Generates a compile-time error
Subclass Static Method	Generated a compile-time error	Hides

8. **If you have N eggs, then you have N/12 dozen eggs, with N%12 eggs left over. (This essentially the definition of the / and % operators for integers.) Write a program that asks the user how many eggs he/she has and then tells the user how many dozen eggs he/she has and how many extra eggs are left over. A gross of eggs is equal to 144 eggs.**

   Extend your program so that it will tell user how many gross, how many dozen, and how many left over eggs he/she has. For example, if the user says that he/she has 1342 eggs, then your program would respond with your number of eggs is 9 gross, 3 dozen, and 10 since 1342 is equal to 9*144 + 3*12 + 10.

   ➔

```
import java.util.Scanner;

class Eggs {
   public static void main (String[] args) {
      Scanner Scan = new Scanner(System.in);
      System.out.print ("How many eggs do you have?\n> ");
      int TotalEggs = Scan.nextInt();
      if (!(TotalEggs >= 144)) {
         System.out.println ("You have " + TotalEggs/12 + " dozen of Eggs and " +
TotalEggs%12 + " Eggs left over.");
      }
      else {
         int GrossEggs = TotalEggs/144;
         int DozenEggs = (TotalEggs-GrossEggs*144)/12;
         int LeftEggs = TotalEggs-GrossEggs*144-DozenEggs*12;
         System.out.println ("Your number of eggs is " + GrossEggs + " gross, " + DozenEggs
+ " dozen, and " + LeftEggs);
      }
      Scan.close();
   }
}
```

9. **Which integer between 1 and 10000 has the largest number of divisors, and how many divisors does it have? Write a program to find the answers and print out the results. It is possible that several integers in this range have the same, maximum number of divisors. Your program only has to print one of them. You might need some hints about how to find a maximum value. The basic idea is to go through all the integers, keeping track of the largest number of divisors that you've seen so far. Also, keep track of the integer that had that number of divisors.**

➔
```
class LrgNumOfDivisors {
  public static void main (String[] args) {
    int CountStore = 1, NumberStore = 1;
    int Count;
    int UpperBound = 10000;
    for (int i=UpperBound; i>=(UpperBound-UpperBound/3); --i) {
      Count = 0;
      for (int j=1; j<=i; ++j) {
        if (i%j == 0)
          ++Count;
      }
      if (Count > CountStore) {
        NumberStore = i;
        CountStore = Count;
      }
    }
    System.out.println ("Number " + NumberStore + " has " + CountStore + " divisors.");
  }
}
```

10. **For this problem, you should write a very simple but complete class. The class represents a counter that counts 0, 1, 2, 3, 4, …. The name of the class should be Counter. It has one private instance variable representing the value of the counter. It has two instance methods: increment() adds one to the counter value, and getValue() returns the current counter value. Write a complete definition for the class, Counter.**

➔
```
class Counter {
  private int CounterValue = 0;

  public void increment() {
    ++CounterValue;
  }

  public int getValue() {
    return CounterValue;
  }
}
```

11. **Suppose that a class, Employee, is defined as follows:**

```
class Employee {
        String lastName;
        String firstName;
        double hourlyWage;
        int yearsWithCompany;
}
```

Suppose that data about 100 employees is already stored in an array:

```
Employee[] employeeData = new Employee [100];
```

Write a code segment that will output the first name, last name, and hourly wage of each employee who has been with the company for 20 years or more.

➔
```
for (int i=0; i<100; ++i) {
  if (employeeData[i].yearsWithCompany >= 20) {
    System.out.println (employeeData[i].firstName + " " + employeeData[i].lastName + "
" + employeeData[i].hourlyWage);
  }
}
```

12. **Write a program that reads one line of input text and breaks it up into words. The words should be output one per line. A word is defined to be a sequence of letters. Any characters in the input that are not letters should be discarded. For example, if the user inputs the line He said, "That's not a good idea."**
then the output of the program should be:

He
said
that
s
not
a
good
idea

➔
```
import java.util.Scanner;
import java.lang.String;

public class BreakString {
  public static void main (String[] args) {
    Scanner Scan = new Scanner(System.in);
    System.out.println ("Enter a string:");
    String input = Scan.nextLine();
    for (int i=0; i<input.length(); ++i) {
      if ((input.charAt(i)>='a' && input.charAt(i)<='z') || (input.charAt(i)>='A' &&
input.charAt(i)<='Z')) {
        System.out.print (input.charAt(i));
      }
      else if (i<input.length()-1) {
```

```
            if ((input.charAt(i+1)>='a' && input.charAt(i+1)<='z') || (input.charAt(i+1)>='A'
&& input.charAt(i+1)<='Z')) {
                System.out.print ("\n" + input.charAt(i+1));
                ++i;
              }
            }
        }
        Scan.close();
    }
}
```

13. Write a program to create a room class, the attributes of this class are room_no, room_type, room_area and ACmachine (Boolean: whether the AC is on or off). In this class the member functions are set_data and display_data. Where set_data is used to set the attribute values and the method display_data is used to display the same.

➔
```
class Room {
    private int room_no;
    private char room_type;              //Type-A,B,C,...
    private double room_area;
    private boolean ACmachine;

    public void set_data (int num, char type, double area, boolean ac) {
        room_no = num;
        room_type = type;
        room_area = area;
        ACmachine = ac;
    }

    public void display_data () {
        System.out.println (room_no + " " + room_type + " " + room_area + " " +
ACmachine);
    }
}

public class CRoom {
    public static void main (String[] args) {
        Room rum = new Room();
        rum.set_data (12, 'B', 32.43, false);
        rum.display_data();
    }
}
```

14. Write a program and create a class 'SimpleObject'. Use a constructor to display the message.

➔
```
class SimpleObject {
  SimpleObject () {
    System.out.println ("Message Displayed");
  }
}

public class MainClass {
  public static void main (String[] args) {
    SimpleObject testObj = new SimpleObject();
  }
}
```

15. Write a program in java to illustrate the use of "static" keyword. Create two methods:

i. Method to multiply two numbers without using static keyword

➔
```
class Multiply {
  private int a;
  private int b;

  public void takeNum (int num1, int num2) {
    a = num1;
    b = num2;
  }

  public int multiplyNum () {
    return a*b;
  }
}

public class NonStatic {
  public static void main (String[] args) {
    Multiply m = new Multiply();
    m.takeNum (5, 2);
    int result = m.multiplyNum();
    System.out.println (result);
  }
}
```

ii. Method to add two numbers using static keyword.

➔
```
class Multiply {
  private static int a = 5;
  private static int b = 2;
```

```
    public int multiplyNum () {
      return a*b;
    }
  }

  public class Static {
    public static void main (String[] args) {
      Multiply m = new Multiply();
      int result = m.multiplyNum();
      System.out.println (result);
    }
  }
```

16. **Write a program to create a class named shape. In this class, we have three subclasses: circle, triangle and square. Each class has two member functions named draw() and erase(). Create these using polymorphism concepts.**

➔
```
  class Shape {
    void draw() {}
    void erase() {}
  }

  class Circle extends Shape {
    void draw () {
      System.out.println ("Circle Drawn");
    }
    void erase() {
      System.out.println ("Circle Erased");
    }
  }

  class Square extends Shape {
    void draw () {
      System.out.println ("Square Drawn");
    }
    void erase () {
      System.out.println ("Square Erased");
    }
  }

  class Triangle extends Shape {
    void draw () {
      System.out.println ("Triangle Drawn");
    }
    void erase () {
      System.out.println ("Triangle Erased");
    }
  }
```

17. What will be the output

```
class Animal {
    //overridden method
    public void display() {
        System.out.println ("I am an animal");
    }
}

class Dog extends Animal {
    //overriding method
    @Override
    public void display() {
        System.out.println ("I am a dog");
    }

    public void printMessage() {
        display();
    }
}

class Main {
    public static void main (String[] args) {
        Dog dog1 = new Dog();
        dog1.printMessage();
    }
}
```

➔ **OUTPUT:** I am a dog