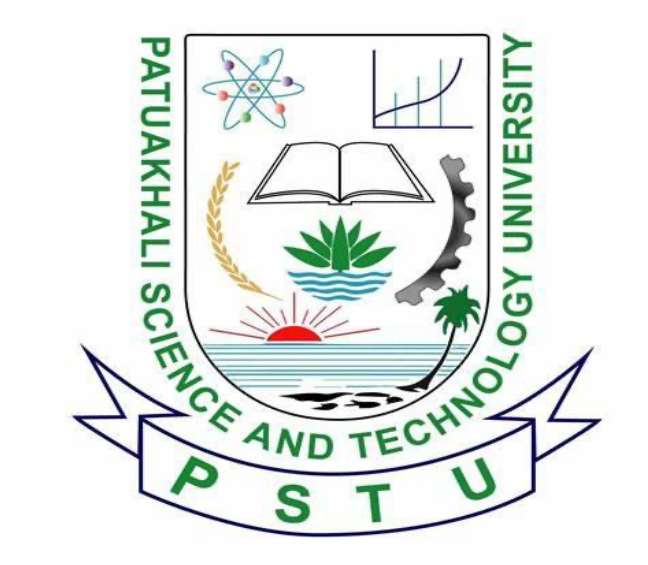
PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY



Course Code: CCE-122

Assignment – 02

SUBMITTED TO:

Sarna Majumder
Department of Computer and Communication Engineering
Faculty of Computer Science and Engineering

SUBMITTED BY:

Name: **MD. MEHEDI HASAN**

ID: **2102016**, Registration No: **10143**

Faculty of Computer Science and Engineering

Polymorphism

input/One

1. Write a console based program to implement polymorphism using inheritance. Consider the example
of Shape as base class with method show(). And then a child class Circle and Rectangle which inherit the
base class Shape and override its method show(). Add one more Method with the name of getInfo().
This method would display the class name in which it is implemented. Do not override this method.
When you will call the method getInfo() with child object it would still show the name of the base class,
which implies that method has been directly inherited and was not overridden.

```
class Shape
{
    void show()
    {
        System.out.println("This is shape class's show");
    }

    void getInfo()
    {
        System.out.println("This is shape class's getInfo");
    }
}

class Circle extends Shape
{
    @Override
    void show()
    {
        System.out.println("This is circle class's show");
    }
}

class Rectangle extends Shape
{
    @Override
    void show()
    {
        System.out.println("This is rectangle class's show");
    }
```

```
}

public class One
{
    public static void main(String[] args) {
        Shape s = new Shape();
        s.show();
        s.getInfo();

        Circle c = new Circle();
        c.show();
        c.getInfo();

        Rectangle r = new Rectangle();
        r.show();
        r.getInfo();
    }
}
```

input/Two

2. Write a subclass called SubClass that is derived from SuperClass and that adds an integer data field
 called data2 and a public method called checkCondition() that will check if data1 is equal to 10 and
 data2 is equal to 15, the checkCondition () method should return “Condition True!”. Also, create
methods called setData2() and getData2() for setting and retrieving the value of data1 and data2, as well
as a constructor that accepts arguments for the starting values of data1 and data2. data1 is data
member of SuperClass.

```
class SuperClass {
    int data1;

    public int getData1() {
        return data1;
    }

    public void setData1(int data1) {
        this.data1 = data1;
    }

}

class SubClass extends SuperClass {
    int data2;
```

```java
    public SubClass(int data1, int data2) {
        super.setData1(data1);
        this.data2 = data2;
    }

    // overloaded constructor
    public SubClass() {
        this(0, 0);
    }

    public String checkCondition() {
        if (super.data1 == 10 && data2 == 15) {
            return "Condition True!";
        }
        return "Condition False!";
    }

    public int getData2() {
        return data2;
    }

    public void setData2(int data2) {
        this.data2 = data2;
    }

    // overriding methods [OPTIONAL]
    @Override
    public void setData1(int data1) {
        super.setData1(data1);
    }

    @Override
    public int getData1() {
        return super.getData1();
    }
}

public class Two {
    public static void main(String[] args) {
        SubClass s = new SubClass(10, 15);
        System.out.println(s.checkCondition());

        s.setData1(0);
        System.out.println(s.getData1());
        System.out.println(s.checkCondition());
    }
}
```

input/Three

3. Create a class named Pizza that stores information about a single pizza. It should contain the
following:
 Private instance variables to store the size of the pizza (either small, medium, or large), the number of
cheese toppings, the number of pepperoni toppings, and the number of ham toppings
Constructor(s) that set all of the instance variables.
Public methods to get and set the instance variables.
A public method named calcCost( ) that returns a double that is the cost of the pizza.
Pizza cost is determined by:
Small: $10 + $2 per topping
Medium: $12 + $2 per topping
Large: $14 + $2 per topping
public method named getDescription( ) that returns a String containing the pizza size, quantity of
each topping.
Write test code to create several pizzas and output their descriptions. For example, a large pizza with
one cheese, one pepperoni and two ham toppings should cost a total of $22. Now Create a PizzaOrder
class that allows up to three pizzas to be saved in an order. Each pizza saved should be a Pizza object.
Create a method calcTotal() that returns the cost of order. In the runner order two pizzas and return the
total cost.

```
class Pizza {
    private String size;
    private int cheeseToppings;
    private int pepperoniToppings;
    private int hamToppings;

    public String getSize() {
        return size;
    }

    public int getCheeseToppings() {
        return cheeseToppings;
    }

    public int getPepperoniToppings() {
        return pepperoniToppings;
```

```
    }

    public int getHamToppings() {
        return hamToppings;
    }

    public void setSize(String size) {
        if (size.equals("small") || size.equals("medium") ||
size.equals("large")) {
            this.size = size;
        } else {
            System.out.println("Invalid size!");
        }
    }

    public void setCheeseToppings(int cheeseToppings) {
        this.cheeseToppings = cheeseToppings;
    }

    public void setPepperoniToppings(int pepperoniToppings) {
        this.pepperoniToppings = pepperoniToppings;
    }

    public void setHamToppings(int hamToppings) {
        this.hamToppings = hamToppings;
    }

    public Pizza(String size, int cheeseToppings, int
pepperoniToppings, int hamToppings) {
        setSize(size);
        this.cheeseToppings = cheeseToppings;
        this.pepperoniToppings = pepperoniToppings;
        this.hamToppings = hamToppings;
    }

    // overloaded constructor
    public Pizza() {
        this("small", 0, 0, 0);
    }

    public double calcCost() {
        double cost = 0;
        switch (size) {
            case "small":
                cost = 10;
                break;
            case "medium":
                cost = 12;
```

```
                break;
            case "large":
                cost = 14;
                break;
        }
        cost += (cheeseToppings + pepperoniToppings + hamToppings) *
2;
        return cost;
    }

    public String getDescription() {
        return "Pizza{" +
                "size='" + size + '\'' +
                ", cheeseToppings=" + cheeseToppings +
                ", pepperoniToppings=" + pepperoniToppings +
                ", hamToppings=" + hamToppings +
                ", cost=" + calcCost() +
                '}';
    }

}

public class Three {
    public static void main(String[] args) {
        Pizza p1 = new Pizza();
        Pizza p2 = new Pizza("medium", 2, 2, 3);
        Pizza p3 = new Pizza("small", 3, 3, 4);
        p3.setSize("large");

        System.out.println(p1.getDescription());
        System.out.println(p2.getDescription());
        System.out.println(p3.getDescription());
    }
}
```

Abstraction

input/Problem_1

1. Create an abstract class 'Parent' with a method 'message'. It has two subclasses each having a method with the same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message' by creating an object for each subclass.

```
abstract class Parent {
    abstract void message();
}

class First extends Parent {
    @Override
    void message() {
        System.out.println("This is First subclass");
    }
}

class Second extends Parent {
    @Override
    void message() {
        System.out.println("This is Second subclass");
    }
}

public class Problem_1 {
    public static void main(String[] args) {
        First first = new First();
        first.message();

        Second second = new Second();
        second.message();
    }

}
```

input/Problem_2

2. Create an abstract class 'Bank' with an abstract method 'getBalance'. $100, $150 and $200
are deposited in banks A, B and C respectively. 'BankA', 'BankB' and 'BankC' are
subclasses of class 'Bank', each having a method named 'getBalance'.

```
Call this method by
creating an object of each of the three classes.

abstract class Bank {
    abstract int getBalance();
}

class BankA extends Bank {
    int balance = 100;

    public int getBalance() {
        return balance;
    }
}

class BankB extends Bank {
    int balance = 150;

    public int getBalance() {
        return balance;
    }
}

class BankC extends Bank {
    int balance = 200;

    public int getBalance() {
        return balance;
    }
}

public class Problem_2 {
    public static void main(String[] args) {
        BankA a = new BankA();
        System.out.println(a.getBalance());

        BankB b = new BankB();
        System.out.println(b.getBalance());

        BankC c = new BankC();
        System.out.println(c.getBalance());
    }
}
```

input/Problem_3

3.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 eac of
the two classes and print the percentage of marks for both the students.

```
abstract class Marks {
    abstract int getPercentage();
}

class A extends Marks {
    int subject_one;
    int subject_two;
    int subject_three;

    public A(int subject_one, int subject_two, int subject_three) {
        this.subject_one = subject_one;
        this.subject_two = subject_two;
        this.subject_three = subject_three;
    }

    @Override
    int getPercentage() {
        return (subject_one + subject_two + subject_three) * 100 /
300;
    }

}

class B extends Marks {
    int subject_one;
    int subject_two;
    int subject_three;
    int subject_four;

    public B(int subject_one, int subject_two, int subject_three,
int subject_four) {
        this.subject_one = subject_one;
        this.subject_two = subject_two;
```

```
        this.subject_three = subject_three;
        this.subject_four = subject_four;
    }

    @Override
    int getPercentage() {
        return (subject_one + subject_two + subject_three +
subject_four) * 100 / 400;
    }

}

public class Problem_3 {
    public static void main(String[] args) {
        A a = new A(100, 80, 65);
        System.out.println(a.getPercentage());

        B b = new B(100, 80, 65, 75);
        System.out.println(b.getPercentage());
    }
}
```

input/Problem_4

```
4. An abstract class has a construtor which prints "This is
constructor of abstract class", an
abstract method named 'a_method' and a non-abstract method which
prints "This is a
normal method of abstract class". A class 'SubClass' inherits the
abstract class and has a
method named 'a_method' which prints "This is abstract method". Now
create an object
of 'SubClass' and call the abstract method and the non-abstract
method. (Analyse the
result)

abstract class Abstract {
    public Abstract() {
        System.out.println("This is constructor of abstract class");
    }

    abstract void a_method();

    void b_method() {
        System.out.println("This is a normal method of abstract
class");
    }
```

```
}

class SubClass extends Abstract {

    @Override
    void a_method() {
        System.out.println("This is abstract method");
    }

}

public class Problem_4 {
    public static void main(String[] args) {
        SubClass subClass = new SubClass();
        subClass.a_method();
        subClass.b_method();
    }
}
```

input/Problem_5

```
5. Create an abstract class 'Animals' with two abstract methods
'cats' and 'dogs'. Now create
a class 'Cats' with a method 'cats' which prints "Cats meow" and a
class 'Dogs' with a
method 'dogs' which prints "Dogs bark", both inheriting the class
'Animals'. Now create
an object for each of the subclasses and call their respective
methods.

abstract class Animals {
    abstract void cats();
    abstract void dogs();
}

class Cats extends Animals {
    @Override
    void cats() {
        System.out.println("Cats meow");
    }

    @Override
    void dogs() {
    }
}
```

```
class Dogs extends Animals {
    @Override
    void cats() {
    }

    @Override
    void dogs() {
        System.out.println("Dogs bark");
    }
}

public class Problem_5 {
    public static void main(String[] args) {
        Cats c = new Cats();
        c.cats();

        Dogs d = new Dogs();
        d.dogs();
    }
}
```

input/Problem_6

6. 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.

```
abstract class Shape {
    abstract void RectangleArea(int length, int breadth);
    abstract void SquareArea(int side);
    abstract void CircleArea(int radius);
}

class Area extends Shape {
    @Override
    void RectangleArea(int length, int breadth) {
```

```
        System.out.println("Area of rectangle: " + length *
breadth);
    }

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

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

public class Problem_6 {
    public static void main(String[] args) {
        Area a = new Area();
        a.RectangleArea(10, 20);
        a.SquareArea(10);
        a.CircleArea(10);
    }
}
```

input/Problem_7

```
// 7. Repeat the above question for 4 rectangles, 4 squares and 5
circles.
// Hint- Use array of objects.

abstract class Shape {
    abstract void RectangleArea(int length, int breadth);
    abstract void SquareArea(int side);
    abstract void CircleArea(int radius);
}

class Area extends Shape {
    @Override
    void RectangleArea(int length, int breadth) {
        System.out.println("Area of rectangle: " + length *
breadth);
    }

    @Override
    void SquareArea(int side) {
        System.out.println("Area of square: " + side * side);
```

```
    }

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

public class Problem_7 {
    public static void main(String[] args) {

        Area[] rectangle = new Area[4];
        rectangle[0] = new Area();
        rectangle[0].RectangleArea(10, 20);

        rectangle[1] = new Area();
        rectangle[1].RectangleArea(15, 18);

        rectangle[2] = new Area();
        rectangle[2].RectangleArea(8, 6);

        rectangle[3] = new Area();
        rectangle[3].RectangleArea(3, 30);
        System.out.println();

        Area[] square = new Area[4];
        square[0] = new Area();
        square[0].SquareArea(10);

        square[1] = new Area();
        square[1].SquareArea(15);

        square[2] = new Area();
        square[2].SquareArea(100);

        square[3] = new Area();
        square[3].SquareArea(1);
        System.out.println();

        Area[] circle = new Area[5];
        circle[0] = new Area();
        circle[0].CircleArea(10);

        circle[1] = new Area();
        circle[1].CircleArea(15);

        circle[2] = new Area();
```

```
        circle[2].CircleArea(13);

        circle[3] = new Area();
        circle[3].CircleArea(1);

        circle[4] = new Area();
        circle[4].CircleArea(3);
    }
}
```

input/Problem_8

8. Create an interface TVremote and use it to inherit another interface smart TVremote.
Create a class TV which implements TVremote interface.

```
interface smartTVremote {
    void powerOn();
}

interface TVremote extends smartTVremote {
    void powerOff();
}

class TV implements TVremote {

    @Override
    public void powerOn() {
        System.out.println("Power is on now");
    }

    @Override
    public void powerOff() {
        System.out.println("Power is off now");
    }

}

public class Problem_8 {
    public static void main(String[] args) {
        TV tv = new TV();
        tv.powerOn();
        tv.powerOff();
    }
}
```