2.0.1 Code Examples: Shape Class

In this section, we will introduce abstract classes, abstract methods, and inheritance. We start with an abstract class Shape. Abstract classes cannot be instantiated. They can only be inherited by child classes.

Listing 1: Shape Class

```
abstract class Shape {
    private int x;
    private int y;
    // constructor
    Shape(int newx, int newy) {
        moveTo(newx, newy);
    }
    // accessor for x & y
    int getX() { return x; }
    int getY() { return y; }
    void setX(int newx) { x = newx; }
    void setY(int newy) { y = newy; }
    // move the x & y position
    void moveTo(int newx, int newy) {
        moveTo(newx);
        moveTo(newy);
    }
    void rMoveTo(int deltax, int deltay) {
        moveTo(getX() + deltax, getY() + deltay);
    }
    // virtual draw method
    abstract void draw();
    abstract double getArea();
    abstract double getPrimeter();
}
```

Rectangle class inherits Shape. Because Shape is an abstract class, Rectangle class has to implement all abstract methods of Shape.

Listing 2: Rectangle Class

```
class Rectangle extends Shape {
    private int width;
    private int height;
```
// constructor
Rectangle(int newx, int newy, int newwidth, int newheight) {
    super(newx, newy);
   setWidth(newwidth);
   setHeight(newheight);
}

// accessor for the width & height
int getWidth() { return width; }
int getHeight() { return height; }
void setWidth(int newwidth) { width = newwidth; }
void setHeight(int newheight) { height = newheight; }

// draw the rectangle
void draw() {
    System.out.println("Drawing a Rectangle at: (" + getX() + ", " + getY() + ", width = " + getWidth() + ", height = " + getHeight() + ");
}

double getArea() {
    return width * height;
}

double getPrimeter() {
    return 2 * (width + height);
}

Circle class inherits Shape. Because Shape is an abstract class, Circle class has to implement all abstract methods of Shape.

Listing 3: Circle Class

class Circle extends Shape {
    private int radius;

    // constructor
    Circle(int newx, int newy, int newradius) {
        super(newx, newy);
        setRadius(newradius);
    }

    // accessor for the radius
    int getRadius() { return radius; }
    void setRadius(int newradius) { radius = newradius; }

    // draw the circle
    void draw() {
        System.out.println("Drawing a Circle at: (" + getX() + ", " + getY() + ", radius = " + getRadius() + ");
    }
```java
    double getArea()
    {
        return radius * radius * Math.PI;
    }
    double getPrimeter()
    {
        return 2 * Math.PI * radius;
    }
```

Square is a kind of Rectangle. Therefore, Square Class inherits Rectangle class. because Rectangle implements the abstract methods of Shape, square does not have to implement them.

Listing 4: Square Class

```java
class Square extends Rectangle{
    public Square(int top , int left , int side){
        super(top, left, side, side);
    }
}
```

Now, you add Triangle and other geometric shapes. Here is the code that test the shape class.

Listing 5: Test Class

```java
public class ShapeTest {
    public static void main(String argv[])
    {
        // create some shape instances
        Shape shapes[] = new Shape[3];
        shapes[0] = new Rectangle(10, 20, 5, 6);
        shapes[1] = new Circle(15, 25, 8);
        shapes[2] = new Square(30, 30, 10);

        // iterate through the list and handle shapes polymorphically
        for (Shape s: shapes){
            s.draw();
            s.rMoveTo(100, 100);
            s.draw();
            double a = s.getArea();
            double p = s.getPrimeter();
            System.out.println("area=" + a + "\tPrimeter=" + p);
        }

        // call a rectangle specific function
        Rectangle arect = new Rectangle(0, 0, 15, 15);
        arect.setWidth(30);
        arect.draw();
        System.out.println("area=" + arect.getArea() + "\tPrimeter=" + arect.getPrimeter());
    }
}
```