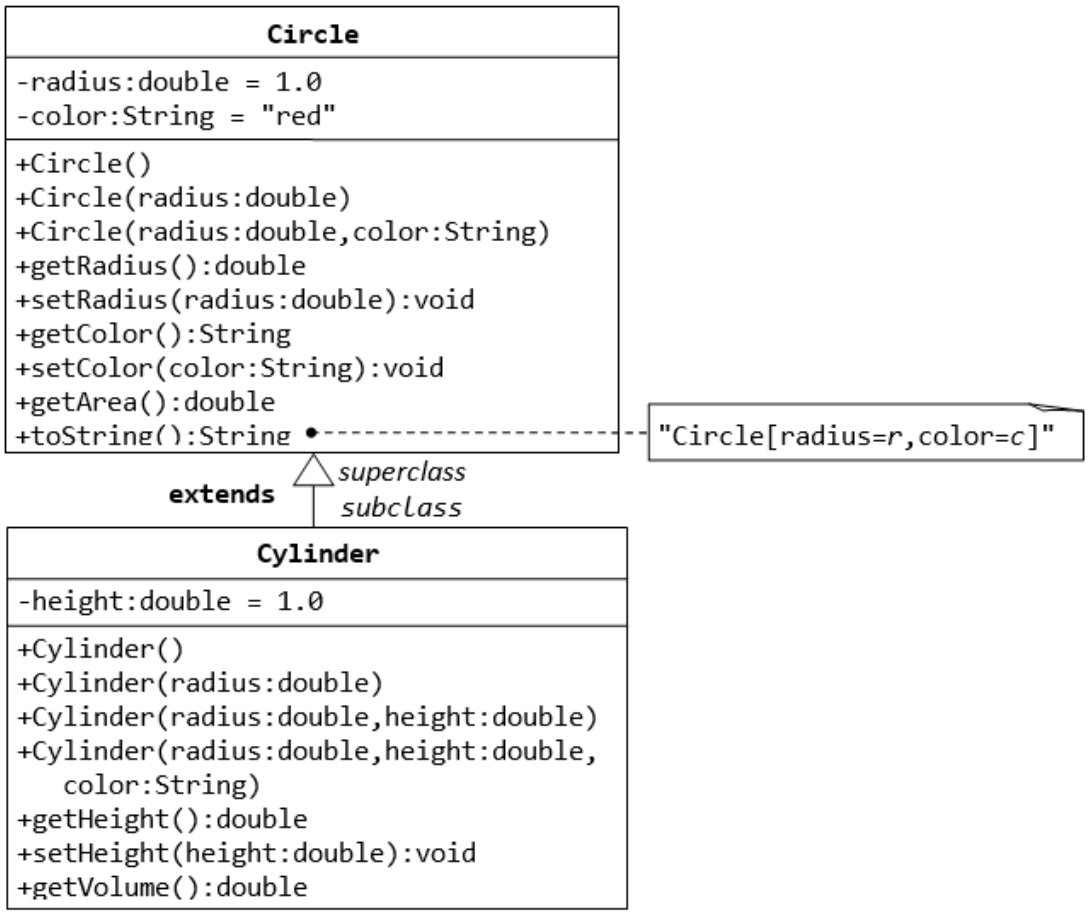
EJERCICIOS DE HERENCIA Y POLIMORFISMO

1. CÍRCULO Y CILINDRO

Escribe la clase `Circle` y la clase `Cylinder`, que hereda de `Circle`. Utiliza el siguiente diagrama de clases para concretar atributos y métodos:



La subclase `Cylinder` debe invocar los constructores de la superclase `Circle` (mediante *super()* y *super(radius))* y hereda los atributos y los métodos de la superclase.

Escribe un programa de prueba para verificar el correcto funcionamiento de las clases definidas.

```java
/*
 * The Circle class models a circle with a radius and color.
 */
public class Circle {  // Save as "Circle.java"
   // private instance variable, not accessible from outside this class
   private double radius;
   private String color;

   // The default constructor with no argument.
   // It sets the radius and color to their default value.
   public Circle() {
      radius = 1.0;
      color = "red";
   }

   // 2nd constructor with given radius, but color default
   public Circle(double r) {
      radius = r;
      color = "red";
   }

   // A public method for retrieving the radius
   public double getRadius() {
     return radius;
   }

   // A public method for computing the area of circle
   public double getArea() {
      return radius*radius*Math.PI;
   }
}
```

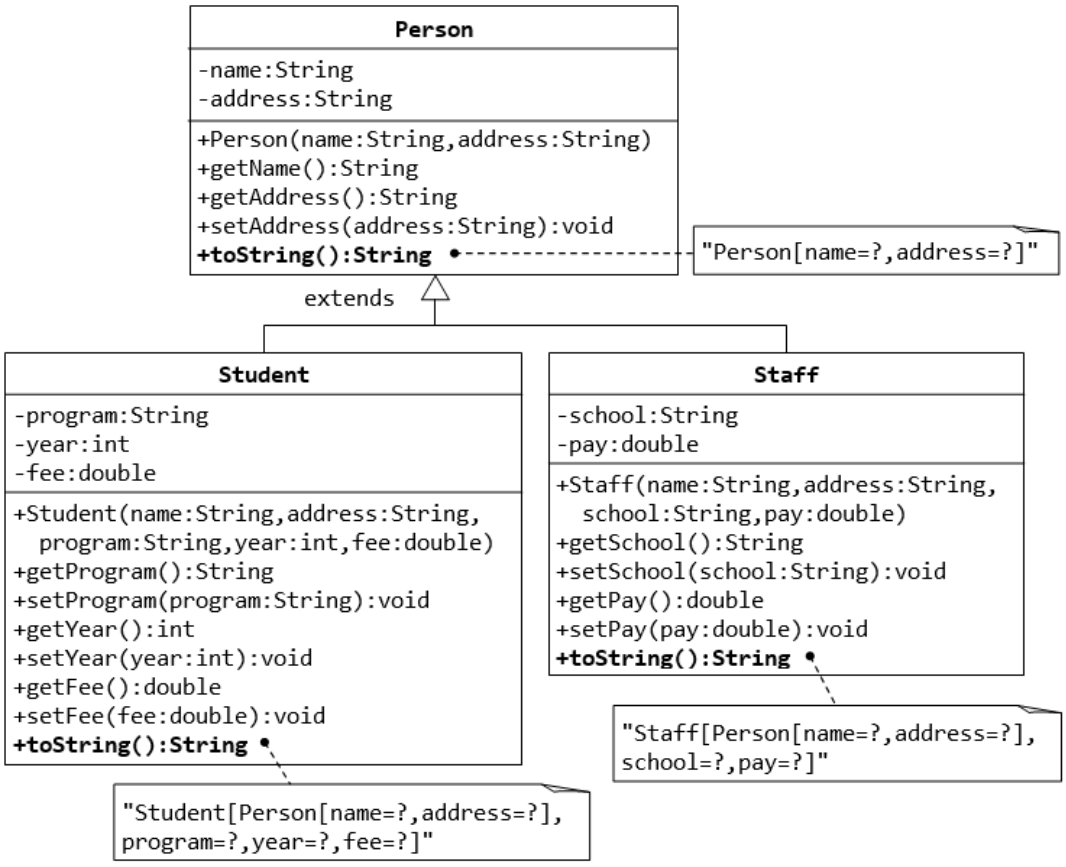
```java
public class Cylinder extends Circle {  // Save as "Cylinder.java"
   private double height;  // private variable

   // Constructor with default color, radius and height
   public Cylinder() {
      super();        // call superclass no-arg constructor Circle()
      height = 1.0;
   }
   // Constructor with default radius, color but given height
   public Cylinder(double height) {
      super();        // call superclass no-arg constructor Circle()
      this.height = height;
   }
   // Constructor with default color, but given radius, height
   public Cylinder(double radius, double height) {
      super(radius);  // call superclass constructor Circle(r)
      this.height = height;
   }

   // A public method for retrieving the height
   public double getHeight() {
      return height;
   }

   // A public method for computing the volume of cylinder
   //  use superclass method getArea() to get the base area
   public double getVolume() {
      return getArea()*height;
   }
}
```

```java
public class TestCylinder {  // save as "TestCylinder.java"
   public static void main (String[] args) {
      // Declare and allocate a new instance of cylinder
      //   with default color, radius, and height
      Cylinder c1 = new Cylinder();
      System.out.println("Cylinder:"
            + " radius=" + c1.getRadius()
            + " height=" + c1.getHeight()
            + " base area=" + c1.getArea()
            + " volume=" + c1.getVolume());

      // Declare and allocate a new instance of cylinder
      //   specifying height, with default color and radius
      Cylinder c2 = new Cylinder(10.0);
      System.out.println("Cylinder:"
            + " radius=" + c2.getRadius()
            + " height=" + c2.getHeight()
            + " base area=" + c2.getArea()
            + " volume=" + c2.getVolume());

      // Declare and allocate a new instance of cylinder
      //   specifying radius and height, with default color
      Cylinder c3 = new Cylinder(2.0, 10.0);
      System.out.println("Cylinder:"
            + " radius=" + c3.getRadius()
            + " height=" + c3.getHeight()
            + " base area=" + c3.getArea()
            + " volume=" + c3.getVolume());
   }
}
```

```java
@Override
public String toString() {      // in Cylinder class
   return "Cylinder: subclass of " + super.toString()  // use Circle's toString()
         + " height=" + height;
}
```

2. PERSONA, ESTUDIANTE Y EMPLEADO

Escribe la clase `Person` y las clases `Student` y `Staff`, que heredan de `Person`. Utiliza el siguiente diagrama de clases para concretar atributos y métodos:



Escribe un programa de prueba para verificar el correcto funcionamiento de las clases definidas.

3. FORMA, CÍRCULO, RECTÁNGULO, CUADRADO

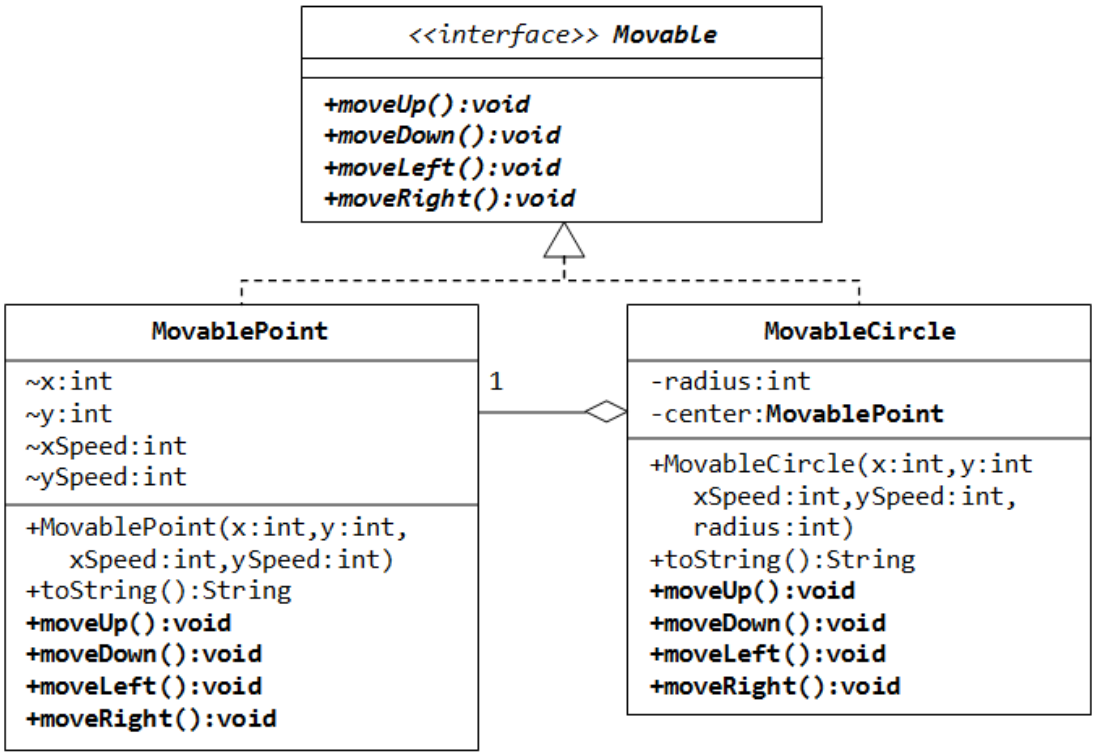
Escribe la clase abstracta `Shape` y sus subclases `Circle`, `Rectangle` y `Square`. Utiliza el siguiente diagrama de clases para concretar atributos y métodos:

Las subclases `Circle` y `Rectangle` deberán sobreescribir los métodos abstractos *getArea()* y *getPerimeter()* y proporcionar una implementación adecuada. Además, deberán sobreescribir el método *toString().*

Escribe un programa de prueba para verificar el correcto funcionamiento que muestre ejemplos de polimorfismo y explica las salidas.

4. MÓVIL, PUNTO MÓVIL Y CÍRCULO MÓVIL

Escribe la interfaz `Movable`, que contiene varios métodos abstractos, y las clases concretas `MovablePoint` y `MovableCircle`, que proporcionan la implementación de dichos métodos de la interfaz. Utiliza el siguiente diagrama de clases para concretar atributos y métodos:



En la clase `MovablePoint`, declara los atributos como variables instancia con acceso de paquete tal como se muestran con `'~'` en el diagrama de clases (las clases del mismo paquete pueden acceder a estos atributos directamente).

En la clase `MovableCircle`, utiliza un `MovablePoint` para representar el centro.

Escribe un programa de prueba para verificar el correcto funcionamiento que incluya las siguientes sentencias:

```
Movable m1 = new MovablePoint(5, 6, 10);
System.out.println(m1);
m1.moveLeft();
System.out.println(m1);

Movable m2 = new MovableCircle(2, 1, 2, 20);
System.out.println(m2);
m2.moveRight();
System.out.println(m2);
```

Explica las salidas producidas por este programa de prueba.