Programación Orientada a Objetos

Especificación para la práctica Geometría

Objetivo:

Aplicar el concepto de Agregación de Clases mediante el diseño e implementación de las clases **`Punto`**, **`Rectángulo`**, y **`Triángulo`** con base en los siguientes lineamientos:

Desarrollo.

Parte 1. Clase Punto y clase Rectángulo.

1. La clase **`Punto`** tiene las coordenadas del plano X-Y, al menos tres constructores, *setter´*s, *getter's*, método **toString**, etc.

```
public class Punto{
    private int x;
    private int y;

   //Ejemplo de constructor de la clase Punto
   public Punto(  ){
      x = y = 0;
   }

public int obtenerX(){
     return x;
}
…
}
//El método toString debe devolver una cadena que contenga el nombre, y los valores de
// X, y Y. Por ejemplo, las sentencias:
 Punto Q = new Punto("Q",3,2);
System.out.println(Q);
// Generan la siguiente salida:
Q(3,2)
// La clase Punto también debe soportar el siguiente código:
Punto M = new Punto("M",7,4);
double d = Q.distancia(M);  // Distancia entre Q y M
d = Q.distancia( ); //Distancia de Q al origen (0,0)
int c = Q.cuadrante( ); // Devuelve  el cuadrante en donde se ubica el punto Q
```

2. Clase **`Rectángulo`**. Cada objeto de la clase **Rectángulo** está definido por dos objetos de la clase **Punto** que denotan la esquina inferior izquierda y la esquina superior derecha.

```
public class Rectangulo{
    private String  nombre;
```

```
    private Punto A;
    private Punto B;

   //Ejemplo de constructor de la clase Rectangulo
   public Rectangulo(  ){
      A = new Punto(1, 1);
      B = new Punto(2,2);
     Nombre = “Incógnito”;
  }
…
}
```

3. Definir al menos tres constructores para Rectangulo:

```
Punto Q = new Punto(“Q”,3,2);
Punto M = new Punto(“M”,7,4);
Punto P = new Punto(“P”,5,3);
Punto N = new Punto(“N”,10,7);
Rectangulo R1 = new Rectangulo(“R1”,Q,M);
Rectangulo R3 = new Rectangulo(“R3”);
Rectangulo R4 = new Rectangulo(R3);
```

4. Definir el método **`union`** que lleva a cabo la unión de dos objetos de tipo **Rectángulo**; el método devuelve un objeto **Rectángulo** que representa la unión.

   Ejemplo de uso:

```
Rectangulo R1 = new Rectangulo(Q,M);
Rectangulo R2 = new Rectangulo(P,N);

Rectangulo U = R1.union(R2);
```

   El objeto U es un rectángulo que representa la unión de los objetos R1 y R2, para este caso las coordenadas de U son: Q(3,2) y N(10,7).

5. Definir el método **`interseccion`** que lleva a cabo la intersección de dos objetos de tipo Rectángulo; el método devuelve un objeto Rectángulo que representa la intersección.

   Ejemplo de uso:

```
Rectangulo R1 = new Rectangulo(Q,M);
Rectangulo R2 = new Rectangulo(P,N);

Rectangulo I = R1.intersección(R2);
```

El objeto I es un rectángulo que representa la intersección de los objetos R1 y R2, para este caso las coordenadas de I son: P(5,3) y M(7,4).

6. Definir el método **`toString()`** para la clase **Rectangulo**:
   Por ejemplo, la siguiente instrucción:

   ```
   System.out.println(R1);
   ```

   Debe dar la siguiente salida en pantalla:

   ```
   //Se invocan el método toString( ) de la clase Rectangulos y los métodos toString de los
   // puntos Q y M
   R1: [Q(3,2), M(7,2)]
   ```

7. Definir el método **`estaAdentro`** que devuelve el valor **`true`** si el objeto de la clase **Punto** que usa como argumento está dentro del área del objeto rectángulo que invoca este método.

   Ejemplo de uso:

   ```
   Rectangulo R1 = new Rectangulo(Q,M);
   Punto X = new Punto(4,3);
   if(R1.estaAdentro(X))
     System.out.println("El punto" + X + "está adentro de " +
   R1);
   //Salida en pantalla:
   El punto X(4,3) está adentro de R1: [Q(3,2), M(7,2)]
   ```

   En este ejemplo, el punto X (4,3) está dentro del área del rectángulo R1. El valor de retorno es por tanto `true`.

8. Implementar el método **`comparar`** el cual compara dos objetos de tipo **Rectángulo**. El método tiene un argumento de tipo **Rectángulo**, y devuelve un entero que indica lo siguiente:

   -1 : El objeto invocante es menor que el objeto del argumento
   0: Los dos objetos son iguales.
   1: El objeto invocante es mayor que el objeto del argumento

   La comparación se realiza con base en el área de los objetos.

   Ejemplo de uso:

   ```
   Rectangulo R1 = new Rectangulo(Q,M);
   Rectangulo R2 = new Rectangulo(P,N);
   ```

```
int v = R1.comparar(R2);
```

En este ejemplo el valor de la variable `v` será de: -1 (el objeto R1 es menor que el objeto R2)

9. Implementar el método **moverRect** que desplaza el objeto de la clase Rectángulo a los dos nuevos puntos dados como argumentos:

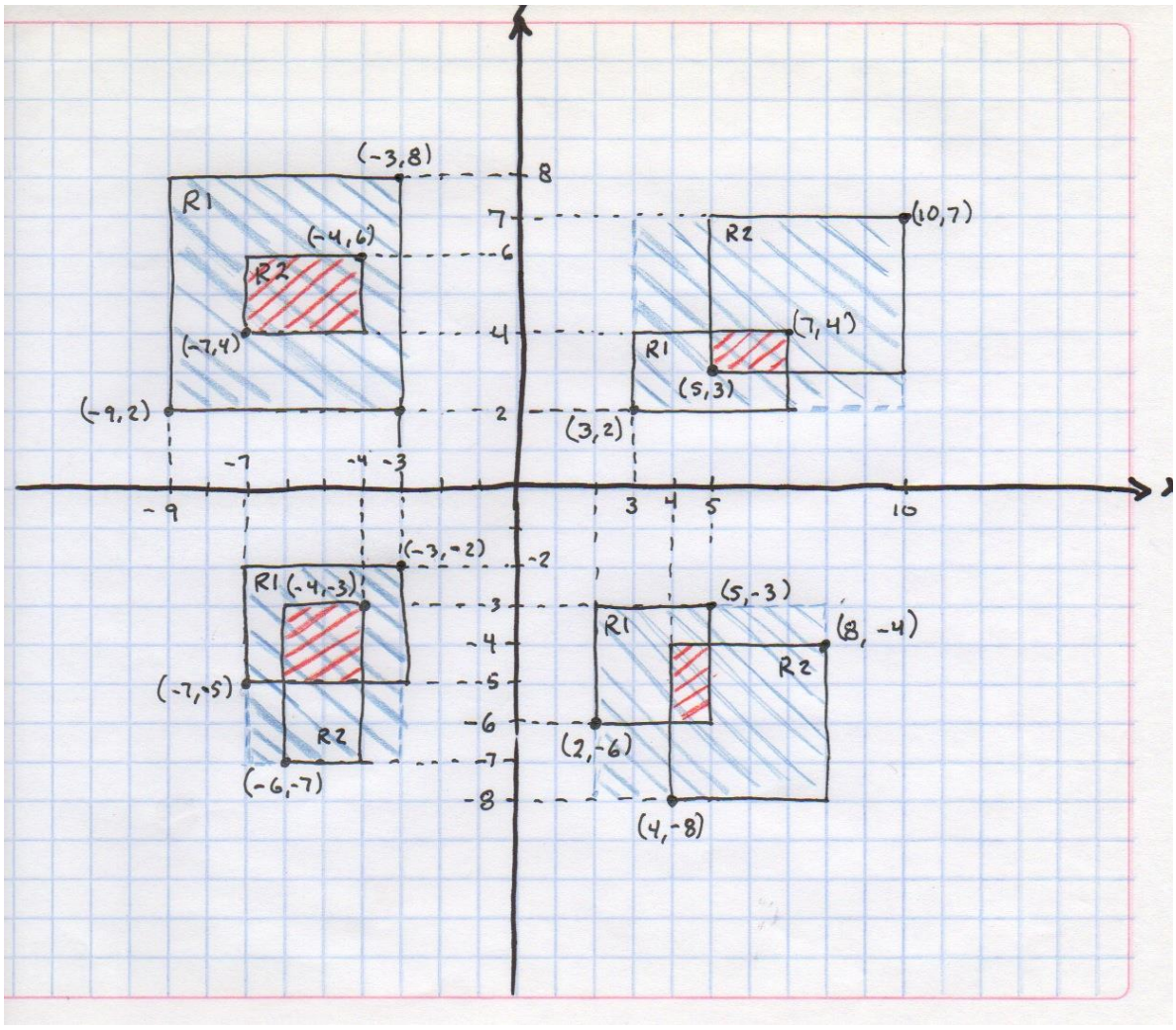
   Punto H = new Punto("H", 20, 30);
   Punto J = new Punto ("J", 30, 50);
   R1.moverRect(H,J);

10. Implementar el método **cuadRect** que devuelve el cuadrante en donde se encuentra el Rectángulo, esto es, el primer punto: esquina superior izquierda.

    int c = R1.cuadRect( );

11. Crear un escenario de prueba en la clase usuaria de la clase Rectángulo de forma que se ilustre el uso de todos los métodos y constructores, para los cuatro cuadrantes del plano cartesiano.

12. Usar como referencia la siguiente figura:
    El sombreado de color rojo representa la intersección; el de color azul la unión.



Parte 2. Clase Triángulo.

1) Realiza todos los puntos, excepto el método **`unión`**, de la Parte 1 para la clase Triángulo. Tú puedes establecer los tipos de triángulos a usar: equiláteros, escalenos o isóceles, o bien todos ellos.
2) Para el método **`intersección`**, el objeto resultante es un objeto Triángulo. Para los casos en que la intersección no sea un triángulo (v.g. un polígono irregular) el método debe devolver **`null`**.
3) Crea el método **`comparar`** tanto para la clase Triángulo como para la clase Rectángulo, de forma que se lleve a cabo la comparación entre: Rectángulo-Rectángulo; Triángulo-Triángulo; Rectángulo-Triángulo; y Triángulo-Rectángulo. Usa el mismo criterio de comparación basado en las áreas de las figuras; así mismo, el método deberá devolver -1, 0, o 1 (menor, igual, mayor).
4) Crear un escenario de prueba en la clase usuaria de la clase Triángulo de forma que se ilustre el uso de todos los métodos y constructores, para los cuatro cuadrantes del plano cartesiano.

Entrega:

- Subir práctica a la plataforma Moodle. Subir sólo archivos con extensión .java, es decir, código fuente. No se aceptan archivos de Netbeans, Eclise o cualquier otro IDE.
- Favor de usar el paquete default de Java.
- Presentar en equipo (máximo dos integrantes) o individualmente.
- Se hará revisión exhaustiva de cada práctica, si se detectan prácticas iguales o muy semejantes se procederá a su cancelación y la calificación será de 0 (cero).

Forma de evaluación:

Concepto	Puntos
Clase Punto	
Constructores y toString	**1**
Método distancia sobrecargado	**1**
Clase Rectángulo	
intersección	**1.5**
unión	**1.5**
estaAdentro y toString	**1**
cuadRect y moverRect	**1**
comparar	**1**
Clase Triángulo	
intersección	**1**
comparar	**1**
Total	**10**

Fecha de entrega:

Viernes 15 de enero de 2020. No hay prórroga.