DECSAI
Departamento de Ciencias de la Computación e I.A.

Complementos de Programación

Convocatoria de Junio. Curso 2012/2013

25 de Junio de 2013

1. **(1 pto)** Explica la ventaja de controlar los posibles errores de ejecución de un programa mediante *excepciones* frente a la forma clásica que se usa en lenguajes como C.

2. **(1 pto)** Indica qué restricción existe a la hora de ordenar múltiples cláusulas `catch` de un bloque `try-catch`, y explica por qué.

3. **(1 pto)** Supongamos las siguientes dos clases:

```
public class ClaseA{
  void metodo1(){
    System.out.println("ClaseA.metodo1()");
  }
}

public class ClaseB extends ClaseA{
  void metodo1(){
    System.out.println("ClaseB.metodo1()");
  }

  void metodo2(){
    System.out.println("ClaseB.metodo2()");
  }
}
```

   Indica qué ocurre con los siguientes programas y explica por qué.

   a) Programa 1

```
public class Programa1{
  public static void main(String args[]){
    ClaseA objetoA;
    ClaseB objetoB=new ClaseB();
    objetoA=objetoB;
    objetoA.metodo1();
  }
}
```

   b) Programa 2

```
public class Programa2{
  public static void main(String args[]){
    ClaseA objetoA;
    ClaseB objetoB=new ClaseB();
    objetoA=objetoB;
    objetoA.metodo2();
  }
}
```

4. **(1 pto)** Explica qué pasa si en una subclase se redefine un **método estático** que estaba definido en su superclase. O sea, el método se implementa en ambas clases con el mismo nombre, con los mismos parámetros, mismo tipo devuelto, pero con un cuerpo diferente.

5. **(2 ptos)** Dado el siguiente código Java, escribe una clase que permita guardar una colección de objetos cuya clase pueda ser cualquier subclase de `Figura`. Debe contener al menos lo siguiente:

   - Un método para añadir un nuevo objeto (de cualquier subclase de `Figura`) a la colección.
   - Un método para recuperar el elemento de la posición `i`.
   - Un método para imprimir en la salida estándar el área de todos los elementos de la colección. Para ello, añade también a las siguientes clases el código necesario para poder obtener el área.

```
abstract class Figura {
  double dim1, dim2;
  Figura(double a, double b) {
    dim1 = a;
    dim2 = b;
  }

}
class Rectangulo extends Figura {
  Rectangulo(double a, double b) {
    super(a, b);
  }

}
class Triangulo extends Figura {
  Triangulo(double a, double b) {
    super(a, b);
  }

}
```

6. **(4 puntos)** Construir un programa en el que se implemente el problema del productor/consumidor, en el que una hebra `Productor` y una hebra `Consumidor` se ejecutan concurrentemente con la hebra principal. La hebra `Productor` escribe números enteros en un buffer y la hebra `Consumidor` lee datos del mismo buffer. Debe asegurarse la correcta sincronización del productor y consumidor en el acceso al buffer compartido. Además, debemos asegurar que cada dato es consumido una sola vez por el consumidor y que el consumidor no consume datos si el buffer está vacío. También debemos asegurar que el productor no produce datos cuando el buffer está lleno. El buffer almacenará los datos enteros en un array. El tamaño del array se determinará con un parámetro del constructor de la clase a la que pertenece el buffer. Como mecanismo de sincronización entre productor y consumidor podrá usarse únicamente los métodos `wait`, `notify` y `notifyAll`.

   Para resolver este problema se construirán las siguientes clases:

   a) Clase `Buffer`: Debe implementarse para que los datos se guarden en un array de enteros (cuya capacidad se definirá mediante un parámetro en el constructor de la clase). Añada además los datos miembro necesarios. No se permite el uso de una *cola bloquante*. Además debe contener un método `get` para obtener el siguiente entero del buffer y un método `set` para añadir un entero al buffer.

   b) Clase `Productor`: Representa la tarea ejecutada por la hebra productora. La función de esta tarea es insertar los números 0 a 99 en el buffer compartido.

   c) Clase `Consumidor`: Representa la tarea ejecutada por la hebra consumidora. La función de esta tarea es obtener la suma de 100 números obtenidos del buffer compartido.

   d) Clase `ProgramaPrincipal`: Contiene la función `main`. Debe crear los objetos necesarios para que funcione el programa y se debe encargar de crear y lanzar las hebras productor y consumidor. El `Buffer` se creará con una capacidad especificada en el primer parámetro de `main`.

Duración del examen: 2 horas y media.