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

Complementos de Programación

Convocatoria de Septiembre. Curso 2013/2014

17 de Septiembre de 2014

1. **(1 pto)** Indica bajo qué condición proporciona Java automáticamente un constructor por defecto
2. **(1 pto)** Muestra la salida del siguiente programa:

```
1  public class Prueba {
2    public static void main(String[] args) {
3      new B();
4    }
5  }
6  class A {
7    int i = 7;
8
9    public A() {
10     setI(20);
11     System.out.println("i en clase A es " + i);
12   }
13   public void setI(int i) {
14     this.i = 2 * i;
15   }
16 }
17 class B extends A {
18   public B() {
19     System.out.println("i en clase B es " + i);
20   }
21
22   public void setI(int i) {
23     this.i = 3 * i;
24   }
25 }
```

3. **(1 pto)** En un bloque `try-catch`, indica qué pasa si ocurre una excepción cuando se está ejecutando una sentencia de la cláusula `catch`
4. **(1 pto)** ¿Para qué sirve la sentencia `throw referenciaaException`?
5. **(3.5 ptos)** Un número racional tiene un numerador y un denominador (enteros `long`) en la forma `a/b`, donde `a` es el numerador y `b` el denominador. Un racional no puede tener `0` como denominador. En la clase que debemos construir, representaremos el numerador y denominador usando números enteros. Escribe la clase `Racional` siguiendo las siguientes indicaciones:

   a) Declara la clase `Racional` como una subclase de la clase `Number` e incluye los datos miembro necesarios. La clase debe implementar también el interfaz genérico `Comparable` para poder comparar dos números racionales. Este interfaz está definido en la biblioteca estándar de Java de la siguiente forma:

   ```
   public interface Comparable<E> {
     public int compareTo(E obj);
   }
   ```

   Recordad que el método `compareTo(E obj)` determina el orden de este objeto respecto al objeto `obj` (parámetro del método) y devuelve un entero negativo, cero o positivo si este objeto es menor, igual o mayor que el objeto `obj` respectivamente.

   La clase `Number` no debe implementarse, pues ya lo está en la biblioteca de Java. La clase `Number` es una clase abstracta que contiene los siguientes métodos públicos:

java.lang.Number
+byteValue(): byte +shortValue(): short ***+intValue(): int*** ***+longValue(): long*** ***+floatValue(): float*** ***+doubleValue(): double***

   Los dos primeros son métodos no abstractos y no necesitan sobreescribirse en la clase `Racional`.

   b) Añade el constructor:

   `Racional()`: Crea un racional con `0` como numerador y `1` como denominador.

c) Incluye un método estático para calcular el máximo común divisor de dos números enteros de tipo `long` siguiendo el algoritmo de Euclides. Según este algoritmo:

- El MCD de a y 0 es igual a a.
- El MCD de a y b es el mismo que el de MCD de b y r, donde r es el resto obtenido al dividir a entre b.

d) Añade el constructor:

`Racional(long numerador, long denominador)`: Crea un racional con el numerador y denominador especificados. Impleméntalo de forma que numerador y denominador queden reducidos a sus términos más pequeños, haciendo uso del máximo común divisor.

e) Sobreescribe el resto de métodos de `Number` (los métodos abstractos). Estos métodos convierten el `Racional` al tipo correspondiente mediante la división del numerador y el denominador.

f) Sobreescribe el siguiente método de la clase `Object` para comprobar si este racional es igual al especificado.

public boolean equals(Object racional2)

g) Sobreescribe el siguiente método de la clase `Object` para devolver un `String` a partir de este racional. El string tendrá la forma `a/b` si `b` es distinto de `1`, y la forma `a` si `b` es igual a `1`.

public String toString()

h) Supongamos que disponemos de un método que lee un número (de alguna de las subclases de `Number`) de la entrada estándar y lo devuelve:

```
public static Number leeNumero()
```

Este método podrá devolver objetos de cualquier subclase de `Number`.

Construir un programa que lea 100 números usando el método anterior y los almacene en un array.

6. **(2.5 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 de 5.

7. **(1 pto)** (Pregunta opcional para subir nota) En la clase `MetodosGenericos`, implementa un método genérico para imprimir en la salida estándar cada uno de los objetos almacenados en un array de un tipo genérico. Indica cómo se usaría este método para imprimir todos los números almacenados en el array construido en el ejercicio 5*h*.

Duración del examen: 2 horas y media.