CC60H : Nuevos Lenguajes de Programación
Control 1–Semestre Otoño'2001
Prof.: Luis Mateu.

Sin apuntes, 1 hora 30 minutos

Pregunta 1

La siguiente es la solución entregada por un programador a un problema en donde se le pide eligir aleatoriamente 100 personas de la lista `listaPersonas` y mostrarla en pantalla en orden alfabético. Encuentre los errores en el programa, explique por qué son errores y corríjalos. Al final de este control encontrá un torpedo con los encabezados de los métodos que se invocan.

```
// ... algunas clases ...
class Persona { String nom, dir; }
class ComparadorPorNombre extends Comparator {
  public static int compare(Persona p1, Persona p2) {
    return p1.nom.compareTo(p2.nom);
  }
}
// ... el programa ...
listaAleatoria= Collections.shuffle(listaPersonas);
lista100= listaAleatoria.subList(0, 100);
listaOrd= Collections.sort(lista100, new ComparadorPorNombre());
for (Iterator it= listaOrd.iterator(); it.hasNext(); it++)
  System.out.println(it.next().nom+" "+it.next().dir);
```

Pregunta 2

En el siguiente programa, en donde B es una subclase de A, se realizan 4 llamadas de métodos sobrecargados :

```
// Las clases
class A {                              | class B extends A {
 void p(int i, int j) { ... }    // 1 |
 void p(byte b, short s) { ... } // 2 |  void p(byte b, short s) { ... }    // 4
 void p(int i, byte b) { ... }   // 3 |
                                      |  void p(short s1, short s2) { ... } // 5
}                                     | }
     // El programa
     byte b; short s;
     ...
     A ab= new B();
     ab.p(b, s);
     ab.p(s, b);
     ab.p(s, s);
     ab.p(b, b);
```

Para cada llamada indique el número del método que :

i. determina estáticamente el compilador para ser invocado.

ii. se invoca finalmente en tiempo de ejecución.

Pregunta 3

La siguiente es la solución entregada por un programador para el problema de la cena de filósofos:

```
class Mesa {                                 |   public void run() {
  int cont= 0;                               |     for(;;) {
  boolean[] tomados= new boolean[5];         |       while (mesa.cont>=4) {
}                                            |         ;
class Filosofo implements Runnable {         |       mesa.cont++;
  Mesa mesa; // Compartida por todos         |       while(mesa.tomados[id])
  int id;    // Identificacion filosofo      |         ;
  Filosofo(int id, Mesa mesa) {              |       mesa.tomados[id]= true;
    this.id= id; this.mesa= mesa;            |       while(mesa.tomados[(id+1)%5])
    new Thread(this).start();                |         ;
  }                                          |       mesa.tomados[(id+1)%5]= true;
                                             |       Comer(id, (id+1)%5);
  void Comer(int i, int j) { ... }           |       tomados[id]= false;
                                             |       tomados[(id+1)%5]= false;
  void Pensar() { ... }                      |       mesa.cont--;
                                             |       Pensar();
                                             | } } }
```

Haga un diagrama de threads mostrando una situación de *deadlock*.

Pregunta 4

Haga las modificaciones que sean estrictamente necesarias para que el programa de la pregunta anterior funcione correctamente, es decir sin *deadlock* y garantizando la exclusión mutua de los threads al tomar cada palito. Justifique por qué es necesario hacer cada modificación.

Torpedo:

```
interface Iterator {
  boolean hasNext();
  Object next();
}

interface List {
  boolean add(Object val);
  boolean add(int idx, Object val);
  Object get(int idx);
  Object set(int idx, Object val);
  Object remove(int idx);
  int size();
  Iterator iterator();
  List subList(int from, int to);
  ...
}

interface Map {
  Object get(Object key);
  Object put(Object key, Object val);
  int size();
  ...
}

interface Comparator {
  int compare(Object o1, Object o2);
}

class Collections {
  static void sort(List l);
  static void sort(List l, Comparator c);
  static void shuffle(List l);
  ...
}

class Integer {
  Integer(int i); // constructor
  int intValue();
  ...
}

class Double {
  Double(double x); // constructor
  double doubleValue();
  ...
}
```