Universidad Carlos III de Madrid

Programación de Sistemas
Grados en Ingeniería de Sistemas Audiovisuales, Ingeniería de Sistemas de Comunicaciones, Ingeniería en Sistemas de Telecomunicación e Ingeniería Telemática

Leganés, 17 de junio de 2013
Duración de la prueba: 45 min.

Examen final. Convocatoria ordinaria. Teoría.
Puntuación: 5 puntos sobre 10 del examen

Sólo una opción es correcta en cada pregunta. Cada respuesta correcta suma 1/4 *puntos. Cada respuesta incorrecta resta* 1/12 *puntos. Las preguntas no contestadas no suman ni restan puntos.*

- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir alguna de estas normas puede ser motivo de expulsión inmediata del examen.
- Marca la respuesta a cada pregunta con una equis ("X") en la tabla de abajo.
- Si marcas más de una opción, o ninguna opción, la pregunta se cuenta como no contestada (ni suma ni resta).
- Rellena **tus datos personales** antes de comenzar a realizar el examen.

Modelo: A

Nombre:	Grupo:

Firma:

NIA:

	A	B	C	D		A	B	C	D
1					11				
2					12				
3					13				
4					14				
5					15				
6					16				
7					17				
8					18				
9					19				
10					20				

1.- Yendo desde la cima (*top*) de la pila hacia el fondo, ¿qué datos contendrá la pila tras ejecutar la secuencia de instrucciones que se muestra?:

```
Stack stack = new Stack();
stack.push("X");
stack.push("Y");
stack.push("Z");
stack.pop();
stack.push("A");
stack.pop();
stack.pop();
stack.push("B");
```

(a) *Y*, *Z*.

(b) *A*, *B*.

(c) *** *B*, *X*.

(d) *B*, *A*, *Z*.

2.- Dado un método *method* que contiene sólo la siguiente llamada recursiva, selecciona de qué tipo de recursión se trata:

```
method(n / 10) + (n % 10)
```

(a) *** recursión lineal

(b) recursión mutua

(c) recursión anidada

(d) recursión en cascada

3.- Si llamamos signatura de un método al número y tipo de parámetros que este recibe, en la reescritura de métodos:

(a) Hay métodos con la misma signatura pero distinto nombre.

(b) Hay métodos con el mismo nombre pero distinta signatura.

(c) *** Hay métodos con el mismo nombre y misma signatura.

(d) Hay métodos con distinta signatura y distinto nombre.

4.- Selecciona la sentencia verdadera:

(a) En una interfaz puede haber métodos ya implementados, en una clase abstracta no.

(b) *** En una clase abstracta puede haber métodos ya implementados, en una interfaz no.

(c) Tanto en las interfaces como en las clases abstractas puede haber algún método implementado.

(d) Ni en las interfaces ni en las clases abstractas puede haber ni un solo método implementado.

5.- En GUIs Java (selecciona la opción *FALSA*):

(a) Los *layout managers* permiten colocar los componentes gráficos.

(b) *** Para crear una ventana, la clase que contiene el método *main* tiene que extender la clase *WindowListener*.

(c) Un objeto *JFrame* no es visible hasta que se invoca su método *setVisible* con argumento *true*.

(d) *JRadioButton* y *JCheckBox* son clases para seleccionar opciones.

6.- Si queremos eliminar de un árbol binario de búsqueda un nodo que tiene dos hijos, ¿cómo hemos de proceder?

(a) Reemplazando el nodo por su hijo.
(b) Un nodo con dos hijos no se puede eliminar.
(c) *** Reemplazando el nodo por el mayor de su subárbol izquierdo o por el menor de su subárbol derecho.
(d) Eliminándolo sin más.

7.- En una cola los datos se extraen:

(a) En un orden que depende de si la implementación se hace sobre un *array* o una lista enlazada.
(b) En orden inverso a aquel en que hayan sido insertados.
(c) *** En el mismo orden en que hayan sido insertados.
(d) En un orden arbitrario.

8.- En Java:

(a) *** No hay herencia múltiple pero sí polimorfismo.
(b) Hay herencia múltiple y polimorfismo.
(c) No hay herencia múltiple ni polimorfismo.
(d) Hay herencia múltiple pero no polimorfismo.

9.- Dada una implementación de lista enlazada cuya clase *Node* contiene únicamente los dos atributos que se muestran a la derecha:

```
public class Node {
    private Object info;
    private Node next;
    (...)
}
```

(a) En un nodo se puede almacenar un objeto de cualquier clase, siempre y cuando dicha clase sea la misma para todos los objetos guardados en todos los nodos de la misma instancia de la lista.
(b) *** En un nodo se puede almacenar un objeto de cualquier clase.
(c) Se puede aprovechar la clase *Node* tal cual se ha definido, sin ningún cambio en sus atributos, para representar un nodo de un árbol binario.
(d) En ningún caso el atributo *next* de un nodo puede tomar valor *null*.

10.- El siguiente método calcula la división entera. Por ejemplo, `division(7, 3)` debería dar como resultado 2. Durante la fase de pruebas, algunos colegas han revisado nuestro código y propuesto varias alternativas para corregir el siguiente método. Selecciona cuál es la correcta:

```
int division (int a, int b) {
   return division(a-b, b) + 1;
}
```

(a) ***

```
int division (int a, int b) {
   if(b > a)
      return 0;
   else
      return division(a-b, b) + 1;
}
```

(b)
```
int division (int a, int b) {
   int res=0;
   if(b <= a)
      division(a-b, b) + 1;
   return res;
}
```

(c) El método original es correcto, no es necesario hacer ningún cambio.

(d)
```
int division (int a, int b) {
   while(a < 0)
      return 0;
   else
      return division(a-b, b) + 1;
}
```

11.- El tamaño de un árbol se calcula:

(a) Moviéndonos recursivamente hacia izquierda o derecha dependiendo del valor de la clave del nodo actual.

(b) *** Sumando el tamaño de todos los subárboles hijos de su raíz, y añadiéndole 1 al resultado.

(c) Hallando el máximo de los tamaños de todos los subárboles hijos de su raíz, y añadiéndole 1 al resultado.

(d) Recorriendo el árbol en pre-orden para imprimir cada nodo.

12.- Implementar una cola doble (*deque*) sobre una lista doblemente enlazada tiene como ventaja con respecto a hacerlo sobre una lista enlazada que:

(a) Permite implementar pilas y colas sobre la cola doble, lo cual no es posible si la cola doble se implementa sobre una lista enlazada.

(b) *** Es más eficiente en términos de tiempo de ejecución en la operación de extracción por el extremo del final de la lista.

(c) Es más eficiente en términos de tiempo de ejecución en la operación de inserción por el extremo del final de la lista.

(d) Su implementación es mucho más sencilla y fácil de entender.

13.- El modificador “final”:

(a) No se puede aplicar a ninguno de ellos, sólo a atributos que son constantes.

(b) Se puede aplicar a métodos, pero no a clases.

(c) *** Se puede aplicar a métodos y clases.

(d) Se puede aplicar a clases, pero no a métodos.

14.- Dada una clase *X* programada para actuar como escuchador de un *JButton*:

(a) La clase *X* tiene que proporcionar el método addActionListener.

(b) La clase *X* tiene que implementar la interfaz ActionEvent.

(c) La clase *X* tiene que proporcionar el método addEventListener.

(d) *** La clase *X* tiene que implementar la interfaz ActionListener.

15.- Sea una lista enlazada con su implementación habitual, cuyo primer nodo es *first*. El valor de la referencia *node* apunta, tras finalizar el bucle en el fragmento de código de la derecha:

```
Node node = first;
while (node.getNext() != null) {
    node = node.getNext();
}
```

(a) Al último nodo de la lista si la lista no está vacía. Si está vacía, apunta a *null*.

(b) *** Al último nodo de la lista si la lista no está vacía. Si está vacía, se lanza una *NullPointerException*.

(c) Al penúltimo nodo de la lista si la lista no está vacía. Si está vacía, apunta a *null*.

(d) A *null* independientemente de que la lista esté vacía o no.

16.- Un árbol ordenado se define como aquel que:

(a) *** Define un orden lineal para cada nodo según el cual ordenar los hijos de éste.

(b) Se puede recorrer en en-orden.

(c) Se puede recorrer en pre-orden.

(d) Se comporta como un montículo.

17.- El algoritmo que extrae y elimina de un montículo el dato con clave menor requiere que se le pase como argumento:

(a) La clave del nodo a eliminar.

(b) El contenido del nodo a eliminar.

(c) La posición del nodo a eliminar.

(d) *** Nada.

18.- Sea un árbol de búsqueda binario implementado de acuerdo a la implementación basada en enlaces que hemos visto en clase. El siguiente método programado en la clase BNode:

```
public void m() {
    System.out.println(info);
    if (left != null)
        left.m();
}
```

(a) *** Imprime la información de los nodos que hay en el camino que va desde este nodo a su descendiente situado más a la izquierda.

(b) Imprime la información del árbol (que tiene a este nodo como raíz) en pre-orden.

(c) Imprime únicamente la información de este nodo y su hijo izquierdo.

(d) Imprime la información de los nodos que hay en el camino que va desde el descendiente situado más a la izquierda hasta este nodo.

19.- Si llamamos signatura de un método al número y tipo de parámetros que este recibe, en la sobrecarga de métodos:

(a) *** Hay métodos con el mismo nombre pero distinta signatura.

(b) Hay métodos con el mismo nombre y misma signatura.

(c) Hay métodos con la misma signatura pero distinto nombre.

(d) Hay métodos con distinta signatura y distinto nombre.

20.- En Java (seleccciona la opción *FALSA*):

(a) Un objeto escuchador puede asociarse a varios componentes gráficos.

(b) Las acciones de usuario se modelan mediante objetos evento.

(c) *** Los contenedores (*containers*) permiten agrupar componentes gráficos que generan el mismo tipo de eventos.

(d) Un escuchador puede modificar componentes gráficos, incluyendo tanto el que genera el evento como otros.