



Representación de Datos Ingeniería Técnica de Telecomunicación – Telemática

Leganés, 17 de septiembre de 2008
Duración de la prueba: 45 min.

No se permite el uso de libros ni apuntes.
Puntuación: 4 puntos sobre 10 del examen.

Cada pregunta tiene una única respuesta correcta. En cada pregunta sólo se puede marcar una opción. La contribución de esta parte a la nota del examen estará entre 0 y 4 puntos. Se calcula dividiendo el número de preguntas acertadas entre 6. **Cada pregunta contestada y fallada resta media pregunta acertada.** Las preguntas no contestadas no restan nada.

Modelo C

1.- En Java, una clase puede heredar (directamente) de:

- (a) Cero, una o varias interfaces.
- (b) Una o varias clases.
- (c) Una clase.

2.- El tipo de recursión del siguiente método:

```
long mor(int n, int m) {  
    if (n == m)  
        return (m + 1);  
    else  
        return mor(n, mor(n - 1, m + 1));  
}
```

- (a) Es *en cascada*.
- (b) Es *no por la cola*.
- (c) Es *anidada*.

3.- Para realizar pruebas de caja negra a un programa:

- (a) Es necesario alcanzar una cobertura del 100 %.
- (b) Es necesario examinar detenidamente el código fuente del programa a probar, y saber qué se supone que debe hacer el programa.
- (c) Hay que saber qué se supone que debe hacer el programa, pero no es necesario examinar su código fuente.

4.- ¿Cuántos datos con su clave asociada puede almacenar una tabla *hash* de 1024 posiciones con resolución de colisiones basada en listas enlazadas?

- (a) Cualquier número de datos, mientras el programa disponga de suficiente memoria.
- (b) Menos de 1024 datos, porque algunas claves tendrán probablemente el mismo valor de *hash*.
- (c) Un máximo de 1024 datos, uno por posición.

5.- Si se ejecuta el método *main* siguiente, en la salida estándar se imprimirá:

```
public class Dato {
    public int n;
}
public class PruebaDato {
    public static void duplicar(Dato dato) {
        dato.n = dato.n * 2;
    }
    public static void main(String[] args) {
        int dato = new Dato();
        dato.n = 5;
        duplicar(dato);
        System.out.println(dato.n);
    }
}
```

- (a) "10".
- (b) "5".
- (c) El código no funciona, porque hacen falta los métodos *getN()* y *setN()* en la clase *Dato*.

6.- Se tiene un método de búsqueda lineal que se ejecuta sobre una colección de N datos, siendo N un valor grande. Se ha comprobado que, cuando se busca un dato que no está en la colección, la operación de búsqueda tarda t unidades de tiempo en finalizar. ¿Cuánto tiempo tardará una búsqueda de un dato que sí esté en la colección de datos?

- (a) t unidades de tiempo, aproximadamente.
- (b) Entre 0 y t unidades de tiempo, aproximadamente.
- (c) $t/2$ unidades de tiempo, aproximadamente.

7.- A continuación se muestra el código de una posible implementación de la operación de inserción de un dato en una *tabla hash*. ¿Qué texto debería aparecer en el lugar marcado con puntos suspensivos?

```
public void insertar(String clave, Object dato) {
    int hash = funcionHash(...);
    tabla[hash] = new NodoHash(clave, dato, tabla[hash]);
}
```

- (a) Nada (se invoca al método sin parámetros).
- (b) Debe aparecer *clave*.
- (c) Debe aparecer *dato*.

8.- Dados los siguientes fragmentos de código, indica qué resultado se mostrará por salida estándar:

```
public class A {
    public String getValor() {return "Hola";}
}
public class B extends A {
    public int getValor() {return 1;}
}
public class Prueba {
    public static void main(String[] args) {
        A ref = new B();
        System.out.println(ref.getValor());
    }
}
```

- (a) "1".
- (b) "Hola".
- (c) No sale nada por salida estándar, porque no compila.

9.- Si tengo un método recursivo por la cola y su equivalente no recursivo que utiliza un bucle, preferiré por razones de eficiencia:

- (a) Da igual a efectos de eficiencia.
- (b) La versión no recursiva.
- (c) La versión recursiva.

10.- La palabra clave *final* se puede aplicar a:

- (a) Sólo a clases.
- (b) Sólo a atributos y métodos.
- (c) Clases, atributos y métodos.

11.- Dada la siguiente expresión aritmética en notación infijo:

$$(3 + (4 - 5)) * (7 / 2)$$

la correspondiente en notación prefijo es:

- (a) 3 4 5 - + 7 2
- (b) * + 3 - 4 5 / 7 2
- (c) + * 3 - 4 5 / 7 2

12.- Una tabla *hash* se utiliza para:

- (a) Ordenar datos.
- (b) Implementar colas con prioridad.
- (c) Buscar datos.

13.- En un montículo completo:

- (a) Todos los niveles están completos.
- (b) Se cumple completamente la propiedad de árbol binario de búsqueda.
- (c) Todos los niveles están completos, excepto posiblemente el último, en donde los nodos están lo más a la izquierda posible.

- 14.- La afirmación “para cualquier método recursivo existe una versión no-recursiva”:
- (a) Es correcta bajo ciertas condiciones.
 - (b) Es siempre incorrecta.
 - (c) Es siempre correcta.
- 15.- Si eliminamos un nodo interno con dos hijos en un árbol binario de búsqueda, podremos sustituirlo por:
- (a) El nodo con clave menor del subárbol de los mayores.
 - (b) El nodo con clave mayor del subárbol de los mayores.
 - (c) El nodo con clave menor del subárbol de los menores.
- 16.- Indica cuál de los siguientes enunciados acerca de los algoritmos de *ordenación por inserción* y *Shell-sort* es correcto:
- (a) Ambos algoritmos tienen un rendimiento muy similar, por lo que es indiferente implementar uno o el otro.
 - (b) Ambos algoritmos se basan en principios de ordenación totalmente distintos.
 - (c) *Shell-sort* consiste en ejecutar varias veces el algoritmo de *ordenación por inserción* con distintos valores de salto.
- 17.- Indica cuál de las siguientes afirmaciones es *falsa*:
- (a) Las pruebas de solidez se utilizan para comprobar que el sistema se comporte adecuadamente cuando se le introducen datos de entrada incorrectos.
 - (b) Las pruebas permiten encontrar errores en un programa, pero no demostrar que el programa no tiene ningún error.
 - (c) Un programa debe empezar a probarse una vez se ha terminado de programar.
- 18.- Un árbol se llama binario, según se ha definido en clase, si:
- (a) Es ordenado y cada nodo tiene 0 ó 2 hijos.
 - (b) Si cada nodo tiene 0 ó 2 hijos (no teniendo que ser ordenado).
 - (c) Si tiene exactamente 2 nodos.
- 19.- Se desea invertir el orden de los elementos de una pila. Esto puede hacerse utilizando exclusivamente:
- (a) Ninguna de las otras dos respuestas es correcta.
 - (b) Un cola auxiliar.
 - (c) Una pila auxiliar.
- 20.- Se dispone de una colección grande de datos, cada uno de ellos asociado a una clave numérica. Los datos no están ordenados. Si se desea realizar una única operación de búsqueda del dato asociado a una clave concreta en la colección, ¿qué forma de realizar la búsqueda resultará, en general, más eficiente?
- (a) Ordenar primero los datos con *quick-sort* y después utilizar búsqueda binaria.
 - (b) Utilizar búsqueda lineal.
 - (c) Utilizar búsqueda binaria.

- 21.- Si tuviera que ordenar una gran cantidad de datos, ¿qué algoritmo preferiría si quiere minimizar el tiempo medio de ejecución?
- (a) Ordenación por inserción.
 - (b) Ordenación por selección.
 - (c) Ordenación *Quick-sort*.
- 22.- Si se declara una clase como:
- ```
public class A implements B {
 (...)
}
```
- (a) La clase *A* hereda los métodos y atributos de la clase *B*, pero no sus constructores.
  - (b) La clase *B* hereda los métodos y atributos de la clase *A*, pero no sus constructores.
  - (c) La clase *A* debe proporcionar una implementación de todos los métodos declarados en la interfaz *B*.
- 23.- Si a un método llamado *prueba* de la clase *MiClase* se lo invoca desde un programa con la sentencia `MiClase.prueba()`, ¿con qué palabra clave debe estar declarado el método *prueba*?
- (a) Con *final*.
  - (b) Con *static*.
  - (c) Con *virtual*.
- 24.- Un determinado algoritmo de ordenación necesita tomar el *array* de datos a ordenar y colocar a un lado del mismo los datos menores que un dato concreto, y al otro los mayores. ¿Cuál es este algoritmo?
- (a) *Shell-sort*.
  - (b) *Quick-sort*.
  - (c) *Merge-sort*.

### Soluciones:

- 1.- c
- 2.- c
- 3.- c
- 4.- a
- 5.- a
- 6.- b
- 7.- b
- 8.- c
- 9.- b
- 10.- c
- 11.- b
- 12.- c
- 13.- c

14.- c

15.- a

16.- c

17.- c

18.- a

19.- b

20.- b

21.- c

22.- c

23.- b

24.- b