Programación de Sistemas
GITT, GIT, GISA, GISC

Leganés, 27 de junio de 2019 — Examen final convocatoria extraordinaria (teoría)
Duración de la prueba: 40 min — Puntuación: 3 puntos sobre 10 del examen

Sólo una opción es correcta en cada pregunta. Cada respuesta correcta suma 0,15 *puntos. Cada respuesta incorrecta resta* 0,05 *puntos. Las preguntas no contestadas no suman ni restan puntos.*

- 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 considera no contestada.
- Rellena **tus datos personales** antes de comenzar a realizar el examen.

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.- Dado el montículo de clave mínima (*min-heap*) resultante de insertar de uno en uno ordenadamente la secuencia de nodos {5, 1, 3, 2, 7}, indica cuál de los siguientes *arrays* se corresponde con el montículo resultante tras realizar una operación `extract()` sobre el montículo construido.

(a) {2, 7, 3, 5}
(b) {2, 3, 5, 7}
(c) {3, 2, 5, 7}
(d) *** {2, 5, 3, 7}

2.- Dado el árbol de búsqueda binario resultante de insertar de uno en uno ordenadamente la siguiente secuencia de nodos {3, 1, 5, 2, 7, 9}, ¿cuál es el resultado de recorrer el árbol en post-orden después de extraer el nodo cuyo valor es 7?

(a) 3, 1, 5, 2, 9
(b) 3, 1, 2, 5, 9
(c) 1, 2, 3, 5, 9
(d) *** 2, 1, 9, 5, 3

3.- Indique qué hace el siguiente método de una lista doblemente enlazada que contiene dos nodos dummy `top` y `tail`.

```
public E m(){
    E result = null;
    if(tail.getPrev()!=top){
      result = tail.getPrev().getInfo();
      tail.setPrev(tail.getPrev().getPrev());
      tail.getPrev().getPrev().setNext(tail);
      size--;
    }
    return result;
}
```

(a) Inserta un elemento en el lado de `top` (`insertFirst`)
(b) *** Elimina un elemento en el lado de `tail` (`removeLast`)
(c) Inserta un elemento en el lado de `tail` (`insertLast`)
(d) Elimina un elemento en el lado de `top` (`removeFirst`)

4.- Dado el siguiente código, indique cuál de las siguientes afirmaciones es cierta.

```
public E m() {
    E result = null;
    if(top!=-1){
        result = data[top];
        data[top] = null;
        top = top -1;
    }
    return result;
}
```

(a) *** Equivale al método `pop()` de un `ArrayStack`

(b) Equivale al método `dequeue()` de una `LinkedQueue`

(c) Equivale al método `removeFirst()` de una `LinkedList`

(d) Equivale al método `push()` de un `ArrayStack`

5.- La resolución de la llamada a un método en caso de sobrecarga depende de...

(a) Si el método es estático o no.

(b) El valor de retorno del método.

(c) *** Los parámetros del método.

(d) La visibilidad del método.

6.- Indique cuál de los siguientes algoritmos de ordenación realiza particiones recursivas eligiendo un elemento como pivote y ordenando mediante intercambios los elementos a la derecha y a la izquierda de este.

(a) Insertion Sort.

(b) *** Quick Sort.

(c) Bubble Sort.

(d) Selection Sort.

7.- Dado el siguiente código, ¿cuál de las siguientes afirmaciones es correcta?:

```
public class ClassA{
    public void m(int a) { ; }
    public void m(float b) { ; }
    public void m(int c) { ; }
    public void m(double d) { ; }
}
```

(a) *** La sobrecarga `public void m(int c)` no es correcta.

(b) La sobrecarga `public void m(double d)` no es correcta.

(c) La sobrecarga `public void m(float b)` no es correcta.

(d) El método `m` está sobrecargado correctamente.

8.- Dado el siguiente código, ¿es correcto el programa?:

```
public class Main{
    public static void main(String args[]){
        int[] elements = {1,2,3,4,5,6};
        int i = 0;
        int len = elements.length;
        while (i<=len){
            System.out.println(elements[i]);
            i++;
        }
    }
}
```

(a) No es correcto, en compilación se indica que no se ha inicializado correctamente el array.

(b) No es correcto, en compilación se indica que no se puede asignar a una variable entera el valor resultante de llamar a `elements.length`.

(c) *** No es correcto, genera una excepción `ArrayIndexOutOfBoundsException` en su ejecución.

(d) Es correcto, en su ejecución imprime por pantalla los elementos del array y finaliza correctamente.

9.- ¿Cuál de las siguientes afirmaciones es correcta?

(a) JUnit muestra el porcentaje de cobertura de las pruebas de caja negra.

(b) Con una cobertura del 100 % del código podemos asegurar el funcionamiento correcto del programa.

(c) *** Las pruebas de caja blanca no aseguran el correcto funcionamiento del programa.

(d) Las pruebas de caja blanca son también llamadas pruebas de entrada/salida.

10.- Dado el siguiente programa, ¿cuál de las siguientes afirmaciones es correcta?:

```
public class Main{
    protected int i;
    public static void main(String args[]){
        Main m = new Main();
        m.i = 0;
        System.out.println(m.i);
    }
}
```

(a) *** Compila y ejecuta correctamente mostrando 0 por pantalla.

(b) Compila y ejecuta correctamente mostrando `null` por pantalla.

(c) No compila correctamente porque no se puede crear un objeto `m` en el método `main` de la propia clase `Main`.

(d) No compila correctamente indicándose que el atributo `i` es protegido y no puede ser accedido por el objeto `m`.

11.- Dado el siguiente código, ¿cuál de las siguientes afirmaciones es correcta?:

```
interface InterfaceA{
    void m();
}
class ClassA implements InterfaceA{
    private int a = 1;
    public void m() {System.out.print(a);}
}
class ClassB extends ClassA implements InterfaceA{
    private int b = 2;
    private int c = 3;
    public void m() {System.out.print(b);}
    public void c() {System.out.print(c);}
```

```
}
public class Main{
    public static void main(String args []){
        InterfaceA ia = new ClassA();
        ia.m();
        ia = new ClassB();
        ia.c();
    }
}
```

(a) El programa muestra por pantalla: 1 2 3..

(b) El programa muestra por pantalla: 1 3.

(c) *** No se puede acceder al método `c()` desde la referencia `ia`.

(d) El programa muestra por pantalla: 1 2.

12.- Dado el siguiente código, ¿cuál de las siguientes afirmaciones es correcta?

```
class ClassA{
    public ClassA() { }
}
class ClassB extends ClassA {
    private int a;
    public ClassB(int a) {super(); this.a = a;}
}
class ClassC extends ClassB{
    private int b;
    public ClassC(int a, int b) {super(); this.b = b;}
}
public class Main{
    public static void main(String args[]){
        ClassC c = new ClassC(10,10);
    }
}
```

(a) Desde un método `static` no se puede crear un objeto.

(b) *** No es correcta la cadena de llamadas de constructores en la herencia cuando se crea el objeto `c`.

(c) En el código de `ClassC` hay que sustituir `super()` por `this()`.

(d) El objeto `c` se crea correctamente.

13.- Dado el siguiente algoritmo recursivo con valores de entrada un array de enteros `a` no vacío y un valor entero `x` a buscar, se podría decir que...

```
public static int binarySearchRecursive(int[] a, int x){
    return binarySearchRecursive(a, 0, a.length-1, x);
}
public static int binarySearchRecursive(int[] a, int first, int last, int x){
    int half;
    if(first <= last){
```

```
            half = (first + last) / 2;
            if (a[half] == x){
                return half;
            }
            else if (a[half]<x){
                return binarySearchRecursive(a, half+1, last, x);
            }
            else if (a[half]>x){
                return binarySearchRecursive(a, first, half-1, x);
            }
        }
    }
```

(a) Es recursivo lineal no por la cola.

(b) Nunca se alcanza el caso base.

(c) *** Es recursivo lineal por la cola.

(d) Es recursivo no lineal anidado.

14.- Indique qué hace el siguiente método en un árbol binario.

```
public int m() {
    int result = 0;
    try {
        result = 1 + getLeft().m() + getRight().m();
    } catch (BTreeException e) {
        result = 0;
    }
    return result;
}
```

(a) Calcula la altura del árbol.

(b) Comprueba si el árbol está vacío y devuelve cero si lo está.

(c) Calcula la profundidad del árbol.

(d) *** Calcula el tamaño del árbol.

15.- Dado el array {3, 1, 7, 2, 5, 9}, ¿cuál sería el contenido del array tras tres iteraciones del bucle externo del método Selection Sort, si usamos la implementación que ordena de menor a mayor eligiendo el valor mínimo de la parte no ordenada e intercambiándolo por el primer elemento desordenado?

(a) {1, 3, 2, 5, 9, 7}

(b) {1, 3, 7, 2, 5, 9}

(c) {1, 3, 2, 5, 7, 9}

(d) *** {1, 2, 3, 7, 5, 9}

16.- Dado el siguiente método al que pasamos como parámetro el array {1, 2, 3}, indica cuál de las siguientes afirmaciones es cierta.

```
public void m(int array[]){
    LStack s = new LStack();
    LQueue q = new LQueue();
    for(int i=0; i<array.length; i++){
        s.push(array[i]);
    }
    for(int i=0; i<array.length; i++){
        q.enqueue(s.pop());
    }
    for(int i=0; i<array.length; i++){
        System.out.println(q.dequeue());
    }
}
```

(a) *** Imprime 3, 2, 1.

(b) No pueden coexistir una pila y una cola en un mismo programa.

(c) Lanza `ArrayIndexOutOfBoundException`.

(d) Imprime 1, 2, 3.

17.- Si compilas este código y *javac* te indica: *error: ClassB is not abstract and does not override abstract method method1() in ClassA*, ¿Qué puedes deducir?:

```
public class Main{
    public static void main(String args[]){
        ClassB b  = new ClassB();
    }
}
```

(a) `ClassB` no hereda de `ClassA` y `ClassB` tiene que ser abstracta.

(b) `ClassB` hereda de `ClassA` y hace falta implementar la interfaz `Abstractable`.

(c) *** `ClassB` hereda de `ClassA`. `ClassA` es abstracta con su método `method1()` también abstracto.

(d) `ClassB` no hereda de `ClassA` pero necesita implementar un método `method1()`.

18.- Indique cuál de las siguientes afirmaciones sobre estructuras de datos es cierta.

(a) Es necesario indicar el tamaño de la lista enlazada antes de su creación.

(b) Las listas enlazadas necesitan disponer de espacio contiguo en memoria para el almacenamiento de los datos que contienen.

(c) Los arrays pueden crecer dinámicamente en función del número de elementos añadidos.

(d) *** Las interfaces `Stack` y `Queue` pueden implementarse tanto con arrays como con listas enlazadas.

19.- Indique cuál de las siguientes afirmaciones es cierta.

(a) *** En un montículo binario mínimo (*min-heap*) los hijos de un nodo siempre tienen una clave mayor que su padre.

(b) Un árbol no se puede representar mediante un array porque es una estructura jerárquica.

(c) Un árbol binario siempre es un árbol de búsqueda.

(d) En un árbol binario, dado un nodo, las claves de los nodos de su subárbol izquierdo siempre son mayores que las de su subárbol derecho.

20.- Dado el siguiente código, ¿cuál de las siguientes afirmaciones es correcta?

```
public class ClassA extends ClassB, ClassC implements InterfaceD, InterfaceE{
    public static void main(String args[]){ }
  }
}
```

(a) El programa no compila porque en `ClassA` no se puede implementar un método `main`.

(b) El programa no compila porque Java no permite la implementación de más de una interfaz al mismo tiempo.

(c) *** El programa no compila porque Java no admite herencia de más de una clase al mismo tiempo.

(d) El programa no compila porque deberíamos haber implementado un constructor de forma explícita.