Grado en Ingeniería Informática	***MODELO B*** Asignatura Estructura de Datos y Algoritmos **27 de Febrero de 2014.** **PRIMER EXAMEN PARCIAL**

Nombre: .....

Apellidos: .....

Grupo: .....

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA

1. Es necesario poner todos los datos del alumno en el cuadernillo de preguntas (este documento). Use un bolígrafo para rellenarlos.
2. El examen de la parte de teoría está compuesto por 10 Preguntas con cuatro posibles respuestas cada una (5 puntos en total).
3. Para que la pregunta multirespuesta se considere contestada correctamente, se deben marcar todas las casillas que sean correctas **(pueden existir una o más contestaciones correctas a cada pregunta, pero nunca ninguna).**
4. Las contestaciones incorrectas **no restan** puntos.
5. Solamente se evaluará la contestación en este cuadernillo de preguntas.
6. Cuando finalice la prueba, se deben entregar el enunciado del examen y cualquier hoja que haya empleado.
7. No está permitido salir del aula por ningún motivo hasta la finalización del examen.
8. Desconecten los móviles durante el examen.
9. La duración del examen es de **30 minutos**.

NO PASE DE ESTA HOJA hasta que se le indique el comienzo del examen

1. En el siguiente código:

```
public class A {

        private String atributo1;
        public String atributo2;

        public A(String atributo1, String atributo2) {

                atributo1=atributo2;
                atributo2=atributo1;
        }
        public void show() {

                System.out.println(atributo1+" "+ atributo2);
        }
        public static void main(String args[]) {

                 A obj=new A();
                obj.show();
        }
}
```

   a) **Hay un error de compilación en el método main porque el constructor sin parámetros no ha sido definido en la clase.**
   b) En el método main, el objeto obj se crea invocando al constructor sin parámetros, que siempre proporciona Java por defecto.
   c) **En el método constructor, se debería haber utilizado la variable this para evitar el conflicto entre los identificadores de los atributos miembro y de los argumentos.**
   d) En el método constructor se actualizan los atributos del objeto a partir de los argumentos del constructor.
   e) Todas las afirmaciones anteriores son falsas.

2. En las siguientes clases:

```
public class A {

        private String atributo1;
        public String atributo2;

        public  void metodo1() {
                System.out.println(atributo1);
        }
        public char metodo2(int i) {
                return atributo1.charAt(i);
        }

}
public class B extends A {

        public void metodo1() {
              System.out.println(atributo1);
        }

        public static void main(String args[]) {
              B obj=new B();
              obj.metodo2(3);
              obj.metodo1();

              obj=new A();
              obj.metodo1();
        }

}
```

   a) B es la clase y A es la subclase que extienda a B. A además de heredar metodo1 de B, también define un nuevo método metodo2 que muestra el carácter en la posición i-esima de la variable atributo1.
   b) En el método main, el polimorfismo permite que la variable obj, declarada como una variable referencia para un objeto de la clase B, pueda ser instanciada una segunda vez como objeto de la clase padre.
   c) **La clase B tiene un error de compilación ya que metodo1 no puede acceder a la variable atributo1 definida como privada en la clase A.**
   d) Todas las respuestas son correctas.

3. En el siguiente código:

```
public abstract class claseAbstracta {

 public int x;
 public  abstract void met1();
public String met2() {
        return String.valueOf(x);
 }

}

public interface interfazA {
        public void metodo1();
        public String metodo2();

}
public class C extends claseAbstracta implements interfazA {
        public String metodo2() {
                return "metodo 2";
        }

}
```

   a) Para que el código sea correcto, el método "met2" debería ser implementado en la clase C.

   b) Para que el código sea correcto basta con definir a la clase C como abstracta.

   c) Para que el código sea correcto, el método "metodo1" debería ser implementado en la clase C.

   d) Todas las afirmaciones anteriores son correctas.

   e) Todas las afirmaciones anteriores son falsas.

4. Dado el siguiente código:

```
public class A {

        private String atributo1;
        public String atributo2;

        public static void metodo1(int x) {
                for (int i=0;i<x;i++) {
                        System.out.println(i);
                }
        }
        public int metodo1(int x) {
                return x+x;
        }
}
```

   a) La segunda ocurrencia del método "metodo1" también debe ser estático porque no maneja ningún miembro de instancia.

   b) El método "metodo1" puede ser sobrecargado porque en el primer caso devuelve void mientras que en el segundo devuelve int.

   c) La sobrecarga no es posible porque ambas ocurrencias presentan el mismo número y tipo de argumentos.

   d) El método "metodo1" puede ser sobrecargado porque en el primer caso se ha definido como static mientras que la segunda vez es definido como un método de instancia.

   e) Todas las afirmaciones anteriores son falsas.

5. En el siguiente código:

```
public class A extends B {
  public int met(int x, String s) {…}
  public int met(int x, String s, double h) {…}
}
```

   a) Si el método met fue definido en B sin parámetros, puede ser sobre-escrito en la clase A, pero nunca sobrecargado añadiendo otros argumentos.

   b) Esos métodos pueden devolver el mismo tipo, si y solo sí, un método es estático y el otro no estático.

   c) La sobrecarga únicamente se admite en métodos heredados de la superclase.

   d) El método "met" es un método sobrecargado.

   e) El método "met" no puede ser un método de B.

6. Describe brevemente la salida que produce la ejecución del método main de la clase A:

```
class B {
    String str = "Hola soy B";
    public String getContenido() { return str; }
}
public class A {
    private B obj = new B();
    public void show () {
        System.out.println (obj.getContenido());
    }
public static void main(String args[]) {
    A objA=new A();
    objA.show();
}
}
```

a) La ejecución del método main de la clase A, imprimirá null, porque el objeto B no ha sido inicializado y por tanto la variable str no tiene valor.

b) El código es incorrecto porque los atributos de un objeto no pueden ser objetos de otra clase.

c) La ejecución del método main de la clase A, imprimirá el mensaje "Hola Soy B" en la consola.

d) El código es correcto si y solo si B es una subclase de A.

e) Todas las afirmaciones anteriores son falsas.

7. Dado el código descrito en la pregunta anterior, si modificamos el método main al siguiente:

```
public static void main(String args[]) {
A objA=new B();
objA.show();
}
```

Razona brevemente si el código es correcto o no.

8. En el siguiente código:

```
public class ClaseA {

        private String atributo1;

        public static void metodo1(int x) {
                for (int i=0;i<x;i++) {
                        System.out.println(atributo1+" " + i);
                }
        }

}
```

a) Hay un error de compilación ya que no es posible referenciar un atributo de objeto desde un método estático.

b) El código es correcto (no dará ningún error de compilación ni lanzará una excepción en tiempo de ejecución), aunque sería más correcto si la variable atributo1 se hubiera definido como s pero debería ser más correcto si el método s fuera estático.

c) Es correcto (no dará ningún error de compilación ni lanzará una excepción en tiempo de ejecución).

d) Cuando se ejecute lanzará una excepción (error) porque no es posible sumar una variable de tipo String (atributo1) y una variable de tipo int (i).

e) Todas las afirmaciones anteriores son falsas.

9. En el siguiente código: public class A extends B implements C {…}
    a) El código es correcto siempre y cuando B y C sean ambas interfaces.
    b) El código es correcto si y solo si B es una clase abstracta y C una interfaz.
    c) El código es correcto siempre y cuando B sea una interfaz y C una clase.
    d) El código es correcto siempre y cuando C sea una interfaz y B una clase.

10. Herencia:
    a) Una subclase hereda únicamente los miembros públicos y aquellos que no tienen ningún modificador de acceso.
    b) Una subclase hereda únicamente los miembros públicos y protected de la clase padre.
    c) La subclase hereda todos los miembros de la clase padre, siempre y cuando estén en el mismo paquete.
    d) Una subclase hereda todos los miembros de la clase padre, excepto aquellos que no tengan ningún modificador de acceso.
    e) **Todas las afirmaciones anteriores son falsas.**