Ayudantía 7:
Ejercicios POO y Java

Profesores: José Luis Martí Lara, Roberto Díaz Urra
Ayudantes:
Hugo Sepúlveda Arriaza
Gabriela Acuña Benito
Lucio Fondón Rebolledo
lucio.fondon@sansano.usm.cl

Universidad Técnica Federico Santa María
Departamento de Informática

Ejercicio 1

¿Cuál(es) de la(s) siguiente(s) afirmación(es) es(son) correcta(s)?

I. Java soporta herencia múltiple.
II. Pueden existir varios métodos con el mismo nombre en una clase, siempre que posean diferentes parámetros.
III.Una interfaz puede extender sólo una clase.
IV. Una clase abstracta sólo puede contener métodos abstractos.

a) Sólo II
b) I y II
c) II y IV
d) I y III

Ejercicio 1

¿Cuál(es) de la(s) siguiente(s) afirmación(es) es(son) correcta(s)?

I. Java soporta herencia múltiple.
II. Pueden existir varios métodos con el mismo nombre en una clase, siempre que posean diferentes parámetros.
III.Una interfaz puede extender sólo una clase.
IV. Una clase abstracta sólo puede contener métodos abstractos.

a) Sólo II
b) I y II
c) II y IV
d) I y III
R: **Alternativa a**

Ejercicio 2 (C2 2016-2)

Dada la siguiente definición de una clase:

```
1 public abstract class Clase {
2     public abstract void f();
3     protected int x;
4 };
```

Indique si las siguientes afirmaciones son verdaderas:

1. ¿Es realmente una clase abstracta?

Ejercicio 2 (C2 2016-2)

Dada la siguiente definición de una clase:

```
1 public abstract class Clase {
2     public abstract void f();
3     protected int x;
4 };
```

Indique si las siguientes afirmaciones son verdaderas:

I. ¿Es realmente una clase abstracta? ✓

II. ¿Se puede derivar de ella?

Ejercicio 2 (C2 2016-2)

Dada la siguiente definición de una clase:

```
1 public abstract class Clase {
2     public abstract void f();
3     protected int x;
4 };
```

Indique si las siguientes afirmaciones son verdaderas:

I. ¿Es realmente una clase abstracta? ✓
II. ¿Se puede derivar de ella? ✓
III. ¿Se pueden crear objetos a partir de ella?

Ejercicio 2 (C2 2016-2)

Dada la siguiente definición de una clase:

```
1 public abstract class Clase {
2     public abstract void f();
3     protected int x;
4 };
```

Indique si las siguientes afirmaciones son verdaderas:

I. ¿Es realmente una clase abstracta? ✓
II. ¿Se puede derivar de ella? ✓
III. ¿Se pueden crear objetos a partir de ella? ✗

Ejercicio 3 (Q3 2019-1)

¿Cuál de las siguientes declaraciones en Java es la **única** correcta?

```java
// a)
interface A {...}
interface B extends A {...}
class C implements A {...}
class D extends B implements C {...}
```

```java
// b)
interface A {...}
interface B extends A {...}
interface C implements A {...}
class D implements B, C {...}
```

```java
// c)
interface A {...}
class B implements A {...}
class C implements A {...}
class D extends B, C {...}
```

```java
// d)
interface A {...}
interface B extends A {...}
interface C extends A {...}
class D implements B, C {...}
```

Ejercicio 3 (Q3 2019-1)

¿Cuál de las siguientes declaraciones en Java es la **única** correcta?

```
1 // a)
2 interface A {...}
3 interface B extends A {...}
4 class C implements A {...}
5 class D extends B implements C {...}
```

```
// b)
interface A {...}
interface B extends A {...}
interface C implements A {...}
class D implements B, C {...}
```

```
1 // c)
2 interface A {...}
3 class B implements A {...}
4 class C implements A {...}
5 class D extends B, C {...}
```

```
// d)
interface A {...}
interface B extends A {...}
interface C extends A {...}
class D implements B, C {...}
```

R: Alternativa d

Ejercicio 4 (C2 2016-2)

Dado el siguiente código en Java, ¿**dónde** y **cómo** invocar el constructor Base desde la clase Sup para que muestre por la salida estándar el mensaje "Hola"? Explicar

```
public abstract class Base{
    Base(int i){
      System.out.println("Hola");
    }

    Base(){
        //...
    }

    public abstract void foo();
}
```

```
public class Sup extends Base{
    Sup(){
        //...
    }

    public void foo() {
        //...
    }

    public static void main(...) {
        Sup s = new Sup();
        //...
    }
}
```

Ejercicio 4 (C2 2016-2)

Dado el siguiente código en Java, ¿**dónde** y **cómo** invocar el constructor Base desde la clase Sup para que muestre por la salida estándar el mensaje "Hola"? Explicar

```
public abstract class Base{
    Base(int i){
      System.out.println("Hola");
    }

    Base(){
        //...
    }

    public abstract void foo();
}
```

```
public class Sup extends Base{
    Sup(){
        //...
    }

    public void foo() {
        //...
    }

    public static void main(...) {
        Sup s = new Sup();
        //...
    }
}
```

R: Es necesario que el constructor de la clase Sup invoque al primero de los dos constructores de la clase Base, con un valor entero como parámetro

Ejercicio 5 (Q3 2017-2)

Considere la siguiente definición de una clase abstracta y una interfaz:

```
1  public abstract class ClasePadre {
2      protected int var1 ;
3      private char var2 ;
4      public ClasePadre (int a, char b){
5          var1 = a; var2 = b;
6      }
7      public int metodo1(){
8          // ...
9      }
10     public abstract void metodo2();
11 }
```

```
public interface Interfaz {
    void metodo1(int a);
    void metodo2(char b);
}
```

Si se desea crear la clase:

```
public class ClaseHijo extends ClasePadre implements Interfaz
```

¿Cuál es el mínimo de elementos que se deben implementar en la clase ClaseHijo para su correcto funcionamiento?

Solución

- `void metodo1(int a)`: Metodo abstracto heredado de `Interfaz`, si bien tiene el mismo nombre que un método implementado en `ClasePadre` posee una firma distinta, por lo tanto debe ser implementado.
- `void metodo2(char b)`: Metodo abstracto heredado de `Interfaz`.
- `void metodo2()`: Metodo abstracto heredado de `ClasePadre`. Este junto con el método anterior deben ser implementados por separado al poseer firmas diferentes.
- `ClaseHijo(int a, char b)`: `ClasePadre`, al poseer un constructor con parámetros, no cuenta con un constructor por defecto, por lo que `ClaseHijo` debe implementar un constructor cuya primera línea sea un llamado al constructor de `ClasePadre` (`super(a,b);`).

Ejercicio 6 (Q3 2018-2)

Considerando las siguientes clases en Java:

```java
class Tipo {
    int a = 1;
    Tipo(){
        a = 3;
    }
    Tipo(int a){
        a = this.a;
    }
    void Print(){
        System.out.println(a);
    }
}
```

```java
class SubTipo extends Tipo {
    int a = 0;
    SubTipo(){
        super(2);
    }
    SubTipo(int b){
        a = b;
    }
    void SubPrint(){
        System.out.println(a);
    }
    public int Metodo(int a) {
        return super.a + this.a + a;
    }
}
```

Pregunta en la siguiente slide

Ejercicio 6 (Q3 2018-2)

Indique qué es lo que imprime en pantalla el siguiente código:

```
1  public class Main {
2      public static void main (String[] args) {
3          SubTipo s1 = new SubTipo();
4          SubTipo s2 = new SubTipo(4);
5          s1.Print();
6          s1.SubPrint();
7          s2.Print();
8          s2.SubPrint();
9          System.out.println(s1.Metodo(5));
10         System.out.println(s2.Metodo(1));
11     }
12 }
```

Ejercicio 6 (Q3 2018-2)

Indique qué es lo que imprime en pantalla el siguiente código:

```
1  public class Main {
2      public static void main (String[] args) {
3          SubTipo s1 = new SubTipo();
4          SubTipo s2 = new SubTipo(4);
5          s1.Print();
6          s1.SubPrint();
7          s2.Print();
8          s2.SubPrint();
9          System.out.println(s1.Metodo(5));
10         System.out.println(s2.Metodo(1));
11     }
12 }
```

R:

```
1
0
3
4
6
8
```