Departamento de Programación
Facultad de Informática
Universidad Nacional del Comahue

Programación Concurrente -2018
Practico General Repaso de JAVA

***Repasemos**:*
Cuando se crea una clase, se crea una plantilla donde se definen ***atributos*** *y* ***métodos.***
Cuando se crea un objeto, ***se instancia una clase,*** *mediante el operador* ***new.***
Todos los objetos de la misma clase tienen los mismos atributos y métodos.
Los valores concretos de cada atributo de cada objeto pueden ser diferentes y conforman su estado.

Ejercicio 1:
Implementar en JAVA dos clases que llamaremos **Suma** y **Resta**. Cada clase tiene como atributo **valor1**, **valor2** y **resultado**. Los métodos a definir son cargar1 (que iniciliza el atributovalor1), cargar2 (que inicializa el atributo valor2), operar (que en el caso de la clase "Suma" suma los dos atributos y en el caso de la clase "Resta" hace la diferencia entre valor1 y valor2.

***Repasemos**:*
La herencia nos permite definir una clase como una ampliación de otra.
Una superclase o clase padre, es una clase que es extendida por otra clase, o que otras clases heredan.
Una subclase, clase hija, es una clase que extiende o amplia a otra clase. Hereda todos los campos y los métodos de la superclase.
La herencia es un mecanismo que nos ofrece una solución al problema de duplicación de código.
La característica esencial de esta técnica es que necesitamos describir las características comunes sólo una vez.
La herencia también se denomina relación «es-un».
La razón de esta nomenclatura radica en que la subclase es una especialización de la superclase.
La herencia nos permite crear dos clases que son bastante similares evitando la necesidad de escribir dos veces la parte que es idéntica.
Más de una subclase puede heredar de la misma superclase y una subclase puede convertirse en la superclase de otras subclases.
Las clases que están vinculadas mediante una relación de herencia forman una jerarquía de herencia.
La herencia es una técnica de abstracción que nos permite categorizar las clases de objetos bajo cierto criterio y nos ayuda a especificar las características de estas clases.
La palabra clave ***extends*** *define la relación de herencia.*
La frase ***«extends <SuperClase>»*** *especifica que esta clase es una subclase de la clase* ***SuperClase.***

Programación Concurrente -2018
Practico General Repaso de JAVA

> ***public class <Nombre SubClase> extends <Nombre SuperClase>***
> ***{…}***
>
> *La subclase define sólo aquellos campos que son únicos para los objetos de su tipo. Los campos de la superclase se heredan y no necesitan ser incluidos en el código de la subclase.*

Ejercicio 2:
Analice el siguiente código, indicando su funcionalidad:

```
import java.util.*;
class Instrumento {
  public void tocar() {
  System.out.println("Instrumento.tocar()");
}

public String tipo() {
 return "Instrumento";
}

public void afinar() {}
}
```

```
class Guitarra extends Instrumento {
  public void tocar() {
  System.out.println("Guitarra.tocar()");
}

public String tipo() { return "Guitarra"; }
  public void afinar() {}
}
```

```
class Piano extends Instrumento {
  public void tocar() {
  System.out.println("Piano.tocar()");
}
public String tipo() { return "Piano"; }
  public void afinar() {}
}
```

Programación Concurrente -2018
Practico General Repaso de JAVA

```
class Saxofon extends Instrumento {
  public void tocar() {
  System.out.println("Saxofon.tocar()");
}
public String tipo() { return "Saxofon"; }
  public void afinar() {}
}
```

```
// Un tipo de Guitarra
class Guzla extends Guitarra {
  public void tocar() {
  System.out.println("Guzla.tocar()");
}
public void afinar() {
  System.out.println("Guzla.afinar()");
}
}
```

```
// Un tipo de Guitarra
class Ukelele extends Guitarra {
    public void tocar() {
      System.out.println("Ukelele.tocar()");
}
public String tipo() { return "Ukelele"; }
}
```

```
public class Musica {
// No importa el tipo de Instrumento,
// seguirá funcionando debido a Polimorfismo:
   static void afinar(Instrumento i) {
   // ...
        i.tocar();
  }

  static void afinarTodo(Instrumento[] e) {
    for(int i = 0; i < e.length; i++)
          afinar(e[i]);
  }
```

Programación Concurrente -2018
Practico General Repaso de JAVA

```
    public static void main(String[] args) {
        Instrumento[] orquesta = new Instrumento[5];
        int i = 0;

        // Up-casting al asignarse el Arreglo
        orquesta[i++] = new Guitarra();
        orquesta[i++] = new Piano();
        orquesta[i++] = new Saxofon();
        orquesta[i++] = new Guzla();
        orquesta[i++] = new Ukelele();
        afinarTodo(orquesta);
    }
} //clase Musica
```

Ejercicio 3:
En un puerto se alquilan amarres para barcos de distinto tipo. Para cada ALQUILER se guardan los datos del cliente, la fecha inicial y la fecha final de alquiler, la posición del amarre (que se indica como un número, por ejemplo amarre 21) y el barco que lo ocupará. Un BARCO se caracteriza por su matrícula, su eslora en metros y año de fabricación. Un alquiler se calcula multiplicando el número de días de ocupación (incluyendo los días inicial y final) por un módulo que se obtiene simplemente multiplicando por 10 los metros de eslora. Ese valor se incrementa en un valor fijo (200 en la actualidad). Tenga en cuenta que ese valor puede variar en el futuro y puede ser distinto en otros puertos.
Sin embargo ahora se pretende diferenciar la información de algunos tipos de barcos:

- Número de mástiles para veleros.
- Potencia en CV para embarcaciones deportivas a motor.
- Potencia en CV y número de camarotes para yates de lujo.

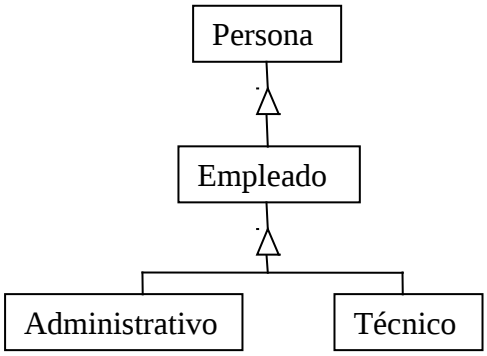
El módulo de los barcos de un tipo especial se obtiene como el módulo normal más:

- El número de mástiles para veleros
- La potencia en CV para embarcaciones deportivas a motor
- La potencia en CV más el número de camarotes para yates de lujo

Utilice la solución propuesta para la materia Prog. Orientada a objetos e impleméntela en Java.
a) Diseñe el diagrama de clases, con detalle de atributos
b) Realice el diagrama de secuencia para el mensaje "calcularValor" de un alquiler. Utilice polimorfismo de forma apropiada. ¿Qué tipo de polimorfismo aplicó?
c) Realice el diagrama de secuencia para el mensaje "registrarAlquiler (nroAmarre, unBarco, unCliente, cantDias)" para registrar un nuevo alquiler en el puerto.

Programación Concurrente -2018
Practico General Repaso de JAVA

Ejercicio 4:
Considere la siguiente jerarquía:



Especifique e implemente las clases Empleado, Administrativo y Técnico. Los datos relevantes de una persona son: nombre, DNI, dirección, fecha de nacimiento y sexo. Un empleado es una persona que desempeña una función en alguna Empresa y recibe un salario por ello. Todos los empleados tienen un legajo y cobran un porcentaje por antigüedad. Un Técnico es un empleado que posee un título y se sabe el año de obtención de su título. Un técnico cobra un adicional según el título que posee. Los empleados administrativos reciben un adicional de acuerdo a su categoría. Utilice la clase persona definida en la materia Programación Orientada a Objetos, implementada en Java, realizando sobre ella los cambios que crea necesarios.

a. Indique de cada clase, variables de instancias y métodos más representativos de la aplicación. Además considere una clase Empresa y grafíquela en el diagrama de clases.
b. Realice un diagrama de secuencia e implementación en Java para la siguiente interacción:
   "Generar una colección con los empleados con antigüedad mayor a 10 años"
c. Utilice polimorfismo para resolver el problema de mostrar por pantalla los datos de cada empleado. Utilice las utilidades del lenguaje y solo implemente lo estrictamente necesario. ¿Qué tipo de polimorfismo aplicó?
d. Considere que se posee una colección (cualquiera) con elementos que tienen la información correspondiente a "título (String)" y el "importe adicional que se cobra por el mismo". En qué clase la colocaría, que tipo de variable sería y qué tipo de método utilizaría para trabajar sobre ella?
e. Realice el diagrama de secuencia para el mensaje cobroMensualEmpleados() que genere una colección cuyos elementos mantengan la información referente al sueldo final que debe cobrar cada empleado. Implemente en Java todos los métodos necesarios
f. Resuelva la siguiente interacción: "Genere una colección ordenada por legajo de todos los empleados con titulo XX".

g. Ahora agregue la siguiente restricción: Cada empleado tiene almacenada la cantidad de horas trabajadas. Si esta es mayor que un máximo de 160 se cobra un plus del 5% del sueldo por horas extras.
h. Los empleados Administrativos además cobran un plus por presentismo, que es del 10% sobre el básico. En el caso de los técnicos, si la cantidad de horas trabajadas es mayor a 100 se cobra un valor fijo de 50$ por hora trabajada adicional hasta las 160 horas.

- Con esta información modifique los métodos que crea convenientes para el cobro de un determinado mes.

PREGUNTAS TEORICAS

Ejercicio 1:

Indicar V o F. Justificar

a. Los Constructores pueden llamar a otros constructores.
b. Se usa this para invocar a un constructor dentro de la clase.
c. El constructor de una clase derivada no puede invocar al constructor de una clase base.
d. Si una clase redefine un método de la clase base, la versión en la clase derivada no afecta a la de la clase base.
e. Siempre se puede crear una instancia de una clase.
f. Una subclase puede tener métodos abstractos.

Ejercicio 2

Responda las siguientes preguntas.

1. Explique cómo se define una clase abstracta en JAVA.

2. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en Java?

3. ¿Cómo se pueden declarar datos que son compartidos por todas las instancias de una clase dada?.

4. Indique cual es el efecto de la aplicación del modificador "final" en:
   a. una variable instancia.
   b. una variable local
   c. un método de instancia
   d. una clase

5. ¿Qué sucede cuando se tiene una variable final en blanco?

Programación Concurrente -2018
Practico General Repaso de JAVA

6. Analice el siguiente trozo de código e indique que sucede

```
package repaso;
public class PruebaStatic {
  public static String aCadena(){
       return "estoy en la superclase";
  }
}

package repaso;
public class RedefinicionStatic extends PruebaStatic {
  public static String aCadena(){
        return super.aCadena() + "estoy en la subclase" ;
  }
}
```

7. ¿Cuál es el alcance de una variable local de un método?

8. ¿Cuando se inicializa un parámetro? ¿Existe una inicialización por defecto?

9. ¿Qué significa hacer un "casting"? ¿Cuando es necesario? ¿En Smalltalk se requiere hacer casting? ¿por qué?

10. ¿Qué sucede cuando declara un método como **protected** en una subclase que está en el mismo paquete que su superclase? ¿Y cuando la subclase esta en otro paquete?

11. Analice el siguiente código:

```
public class Padre {
   public void hacerAlgo() {...}
}

public class Hijo extends Padre {
  private void hacerAlgo() {...}
}
```

Programación Concurrente -2018
Practico General Repaso de JAVA

```
public class Usa {
    public void hacerCosas(){
       Padre p = new Padre();
       Padre h = new Hijo();
       p.hacerAlgo();
       h.hacerAlgo();
    }
}
```

a. ¿Qué problema encuentra?
b. ¿Cómo lo resuelve?

12. Considere las clases PruebaUno y RedefinicionUno dadas a continuación:

```
package repaso;
  public class PruebaUno {
    protected int temp;
    public PruebaUno(int valor) {
       temp = valor;
    }
    public void cambiar(int valor){
         temp = valor;
    }
    public String aCadena(){
         return "estoy en la superclase con valor " + temp;
     }
  }

 package repaso;
public class RedefinicionUno extends PruebaUno {
  private int temp2;
  public RedefinicionUno(int valor){
      super(valor);
      temp2 = valor * 7;
  }
  public String aCadena(){
     return "ahora estoy en la subclase con valor: " + temp ;
  }
  public String aCadena2() {
     return "en la subclase con valor: " + temp2;
   }
}
```

Programación Concurrente -2018
Practico General Repaso de JAVA

a. ¿Qué tipo de polimorfismo se utiliza en el método aCadena? ¿Cómo puede mejorarlo?
b. Qué pasa si el paquete de RedefinicionUno fuera prueba y no repaso?

c. Analice el siguiente código. Identifique errores y corrijalos.

```
package repaso;
  public class TestPruebas {
    public static void main(String[] args) {
          PruebaUno aux[]= new RedefinicionUno[3];
          aux[0]= new PruebaUno(4);
          aux[1]= new RedefinicionUno(4);
          aux[2]= new PruebaUno(14);
          for ( PruebaUno elem: aux)
                System.out.println(elem.aCadena());
     }
}
```

- Considere ahora el siguiente código para el main. ¿Qué sucede? ¿Cómo lo resuelve? Justifique

```
public static void main(String[] args) {
       PruebaUno aux[]= new PruebaUnoUno[3];
         aux[0]= new PruebaUno(4);
         aux[1]= new RedefinicionUno(4);
         aux[2]= new PruebaUno(14);
         for ( PruebaUno elem: aux)
         System.out.println(elem.aCadena());
         for ( PruebaUno elem:aux)
         System.out.println(elem.aCadena2());
}
```