IPC1

Vacaciones Diciembre 2017

Agenda

- Conceptos POO
- Fundamentos de POO
- Manejo de memoria
- Casteo de datos
- Constructores
- Tarea 3, Tarea 4, Tarea Práctica 1

Programación orientada a Objetos (POO)

Cuando se escribe un programa en un lenguaje orientado a objetos, definimos una plantilla o clase que describe las características y el comportamiento de un conjunto de objetos similares.

Una clase es por tanto una plantilla implementada en software que describe un conjunto de objetos con atributos y comportamiento similares.

Una instancia u objeto de una clase es una representación concreta y específica de una clase y que reside en la memoria del ordenador.

Atributos

Los atributos son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en variables denominadas de instancia, y cada objeto particular puede tener valores distintos para estas variables.

Las variables de instancia también denominados miembros dato, son declaradas en la clase pero sus valores son fijados y cambiados en el objeto.

Además de las variables de instancia hay variables de clase, las cuales se aplican a la clase y a todas sus instancias. Por ejemplo, el número de ruedas de un automóvil es el mismo cuatro, para todos los automóviles.

Comportamiento

El comportamiento de los objetos de una clase se implementa mediante funciones miembro o métodos. Un método es un conjunto de instrucciones que realizan una determinada tarea y son similares a las funciones de los lenguajes estructurados.

Del mismo modo que hay variables de instancia y de clase, también hay métodos de instancia y de clase. En el primer caso, un objeto llama a un método para realizar una determinada tarea, en el segundo, el método se llama desde la propia clase.

Fundamentos POO

- Encapsulamiento
- Herencia
- Abstracción
- Polimorfismo

Encapsulamiento (I)

El encapsulamiento habla del modo de ocultar como ha sido implementado el estado, los atributos, de un objeto. Se accede a este estado a través de los métodos públicos. Una buena práctica es hacer las validaciones correspondientes a los posibles estados del objeto, en estos métodos, de modo tal de mantener al objeto en un estado consistente.

También se lo llama “information hiding”. De la misma forma podemos respetar el encapsulamiento si se tiene, en la clase Auto, un atributo velocidad, que sea privado, el único modo de modificar la velocidad es a través de los método acelerar() y frenar(), es decir que esta encapsulada la velocidad, y solo se la puede modificar por los métodos acelerar() y frenar(), no se puede cambiar la velocidad de ninguna otra forma.

Encapsulamiento (II)

```
public class Persona{


        //Atributos
        prívate int altura;


        //Métodos
        public int getAltura(){
        return this.Altura;
        }

        public void setAltura(int unaAltura){
        this.altura = unaAltura;
        }


}
```

Encapsulamiento (III)

Métodos de acceso: Son el medio de acceder a los atributos privados del objeto.

-El getter El método para acceder a los atributos en forma de solo lectura se los denomina “getters”. Son los métodos que retornan el valor de los atributos. El NetBeans, como la mayoría de los entornos de desarrollo, permite generarlos de forma automática.

-El setter: El método para acceder a los atributos en forma de escritura se los denomina “setters”. Son los métodos que establecen el valor de los atributos. También se los genera de forma automática en los entornos de desarrollo.

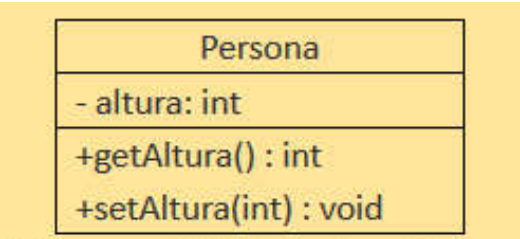
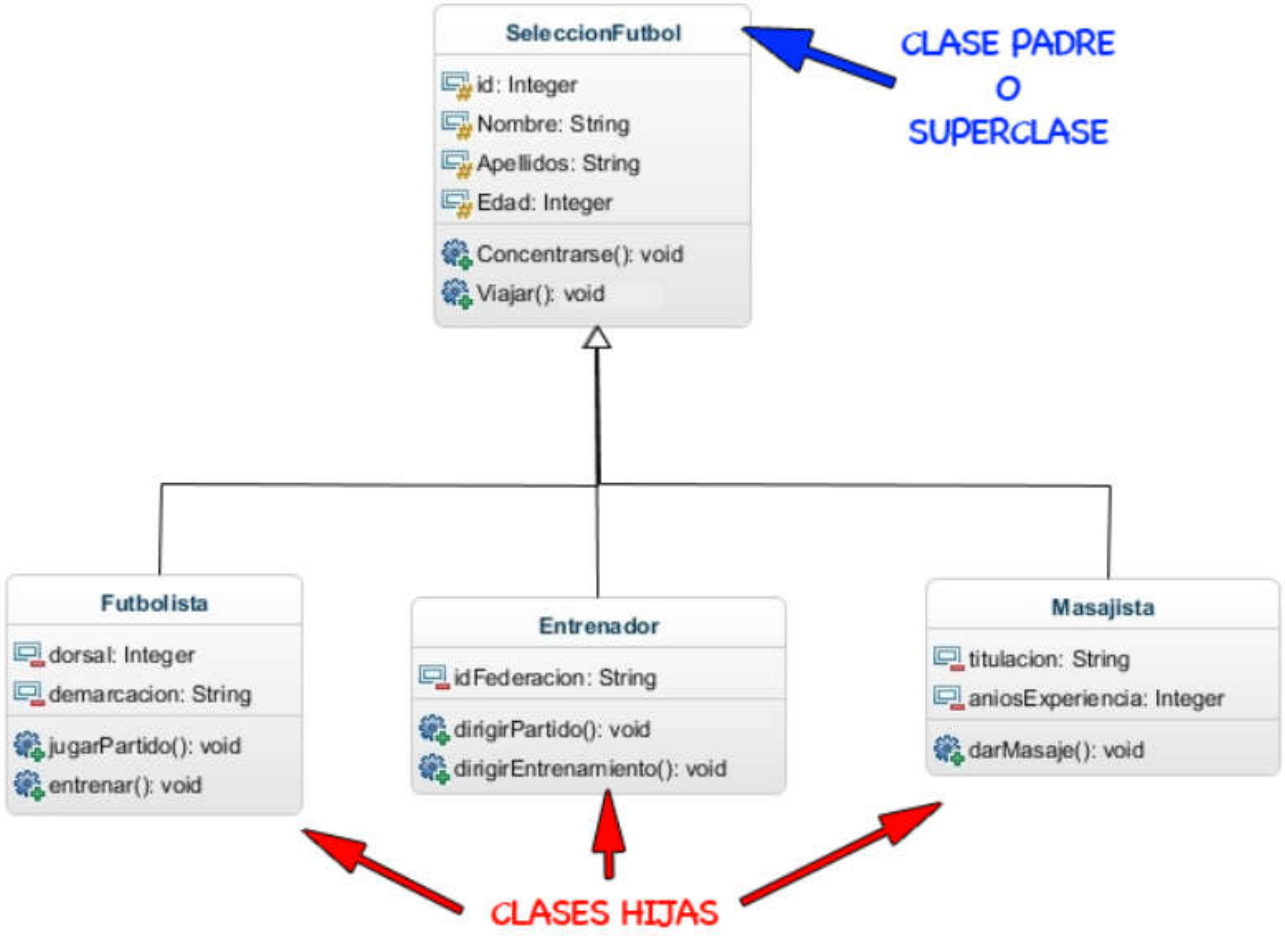


*Diagrama UML de la clase Persona

Herencia (I)

La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente.

La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos.



Herencia (II)

Extends: Esta palabra reservada, indica a la clase hija cual va a ser su clase padre.

Protected: sirve para indicar un tipo de visibilidad de los atributos y métodos de la clase padre y significa que cuando un atributo es protegido, solo es visible ese atributo o método desde una de las clases hijas y no desde otra clase.

Super: sirve para llamar al constructor de la clase padre.

Herencia (III)

```java
public class SeleccionFutbol {

        ......

        public SeleccionFutbol() {
        }

        public SeleccionFutbol(int id, String nombre, String apellidos, int edad) {
                this.id = id;
                this.Nombre = nombre;
                this.Apellidos = apellidos;
                this.Edad = edad;
        }
```

```java
public class Futbolista extends SeleccionFutbol {
        ......
        public Futbolista() {
                super();
        }

        public Futbolista(int id, String nombre, String apellidos, int edad, int dorsal, String demarcacion) {
                super(id, nombre, apellidos, edad);
                this.dorsal = dorsal;
                this.demarcacion = demarcacion;
        }
```

Abstracción

La abstracción es como se pueden representar los objetos en modo de código.

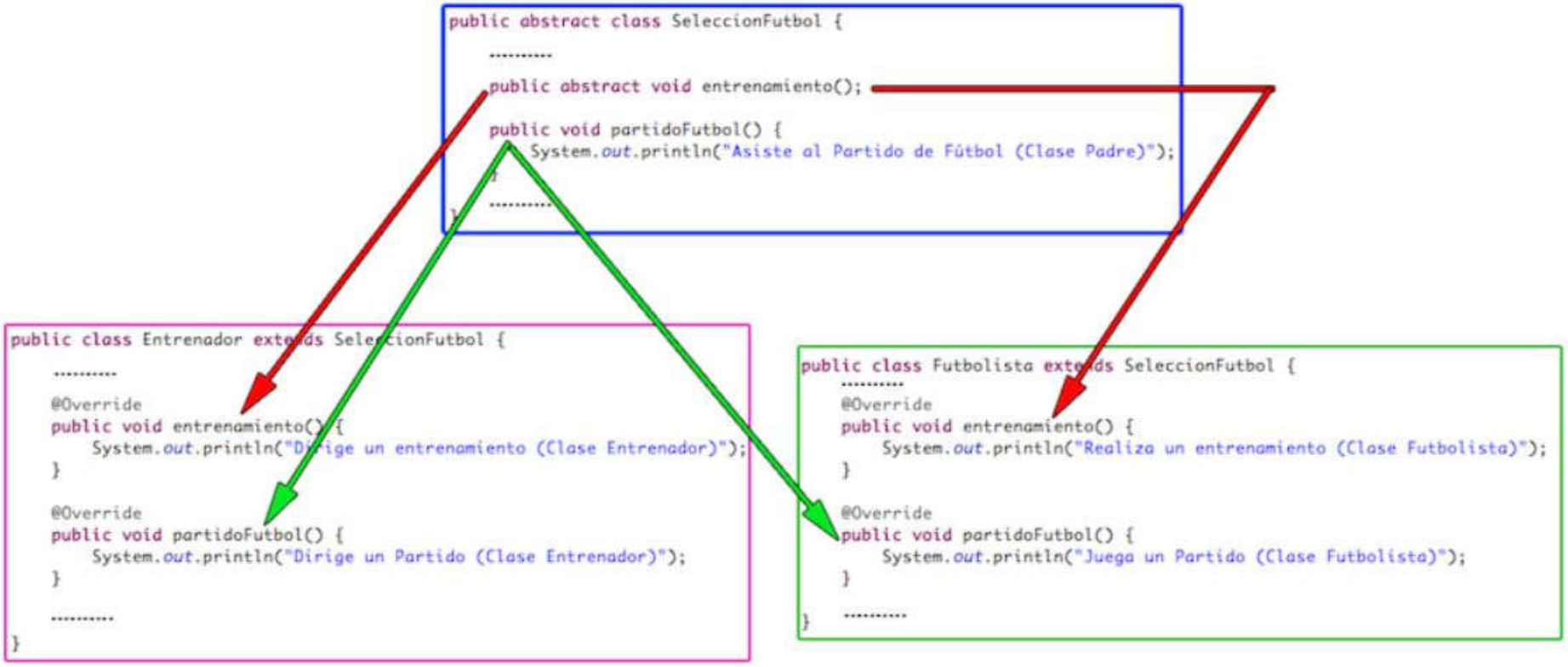
Es un método por el cual abstraemos, vale la redundancia, una determinada entidad de la realidad; sus características y funciones que desempeñan. Estos son representados en clases por medio de atributos y métodos de dicha clase.

Un ejemplo sencillo para comprender este concepto seria la abstracción de un Automóvil.

Acá vamos a sacar de esta entidad sus características por ejemplo: color, año de fabricación, modelo, etc. Y ahora sacamos sus métodos o funciones típicas de esta entidad como por ejemplo: frenar, encender, etc. A esto se le llama abstracción.

Polimorfismo

Es la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos.



Manejo de memoria

ZONA DE DATOS

Es donde se almacenan las instrucciones del programa, las clases con sus métodos y constantes (menos los finals). Esta zona de memoria es fija, y no se puede modificar durante el tiempo de ejecución.

STACK

El tamaño del Stack se define durante el tiempo de compilación y es estático durante su ejecución, por lo que puede llegar un momento en el que lo llenásemos y obtuviésemos un StackOverflow que en java se representa mediante un 'OutOfMemoryException' . Los datos que se almacenan aquí son las referencias a objetos (instancias de objetos) y los datos primitivos como int, float o char. Cuando ejecutamos un método con variables locales, estas se cargan en el Stack y se eliminan una vez se finaliza el método.

HEAP

El Heap es la zona de memoria dinámica, almacena los objetos que se crean, en un principio tiene un tamaño fijo asignado por la JVM (Java Virtual Machine), pero según es necesario se va añadiendo más espacio.

Garbage Collector

Es un proceso de baja prioridad que se ejecuta en la JVM y es el encargado de liberar la memoria que no se emplea. El ser de baja prioridad supone que no pueda estar todo el rato trabajando, y que solo se le asigne su tarea cuando el procesador no tiene un trabajo con mayor prioridad en ejecución.

¿Cómo sabe el Garbage Collector lo que puede borrar y lo que no? Es algo muy simple, si un objeto no tiene referencias desde el Stack tiene que ser eliminado.

Entre sus contras tenemos que al tratarse de un proceso de prioridad baja, es poco probable que se ejecute cuando se esté haciendo un uso intensivo de la CPU.

Casteo de datos

El casteo (casting) es un procedimiento para transformar una variable primitiva de un tipo a otro, o transformar un objeto de una clase a otra clase siempre y cuando haya una relación de herencia entre ambas.

Existen distintos tipos de casteo (casting) de acuerdo a si se utilizan tipos de datos o clases.

- Casteo Implícito
- Casteo Explicito
- Upcasting

Casteo Implícito

El casteo implícito radica en que no se necesita escribir código para que se lleve a cabo. Ocurre cuando se realiza una conversión ancha – widening casting – es decir, cuando se coloca un valor pequeño en un contenedor grande.

```
//Define una variable de tipo int  con el valor 100
int numeroEntero = 100;
//Define una variable de tipo long a partir de un int
long numeroLargo = numero;
```

Casteo Explicito

El casteo explicito se produce cando se realiza una conversión estrecha – narrowing casting – es decir, cuando se coloca un valor grande en un contenedor pequeño. Son susceptibles de perdida de datos y deben realizarse a través de código fuente, de forma explicita.

```
//Define una variable del tipo int con el valor 250
int numeroEntero = 250;
//Define una variable del tipo short y castea la variable numero
short s = (short) numero;
```

Upcasting

El upcasting se produce a nivel objetos. Suponiendo que existe una clase Empleado y clase Ejecutivo, la cual es una subclase de esta. Un Ejecutivo entonces es un empleado.

```
//Instancia un ejecutivo en una variable de tipo Empleado
Empleado e1 = new Ejecutivo ("Maximo Dueño", 2000);
```

```
//Instancia un ejecutivo en una variable de tipo Empleado
Empleado emp = new Ejecutivo ("Máximo Dueño", 2000);

//Se convierte (o castea) la referencia de tipo
Ejecutivo ej = (Ejecutivo) emp;

//Usamos el método de la clase Ejecutivo
ej.ejecutarPlanificacion();
```

Constructores (I)

Un objeto de una clase se crea llamando a una función especial denominada constructor de la clase. El constructor se llama de forma automática cuando se crea un objeto, para situarlo en memoria e inicializar los miembros dato declarados en la clase. El constructor tiene el mismo nombre que la clase. Lo específico del constructor es que no tiene tipo de retorno.

```
class Rectangulo{
        int x;
        int y;
        int ancho;
        int alto;
   Rectangulo(int x1, int y1, int w, int h){
        x=x1;
        y=y1;
        ancho=w;
        alto=h;
   }
}
```

Constructores (II)

El constructor recibe cuatro números que guardan los parámetros *x1*, *y1*, *w* y *h*, y con ellos inicializa los miembros dato *x*, *y*, *ancho* y *alto*.

Una clase puede tener más de un constructor. Por ejemplo, el siguiente constructor crea un rectángulo cuyo origen está en el punto (0, 0).

```
class Rectangulo{
        int x;
        int y;
        int ancho;
        int alto;
   Rectangulo(int w, int h){
        x=0;
        y=0;
        ancho=w;
        alto=h;
   }
}
```

Constructores (III)

Este constructor crea un rectángulo de dimensiones nulas situado en el punto (0, 0)

```
class Rectangulo{
        int x;
        int y;
        int ancho;
        int alto;
    Rectangulo(){
        x=0;
        y=0;
        ancho=0;
        alto=0;
    }
}
```

Tarea 3, Tarea 4, Tarea práctica 1

Hacer el código java para el método que nos indique si un jugador ha Ganado en diagonal, tomando en cuenta que nuestra matriz lleva el nombre de MAT.

Entrega 19/12/2017 antes de las 22:00 por correo asunto IPC1_TareaPractica1 enviar el método java en un pdf con nombre TareaPractica1_#carné