1 Conceptos de programación orientada a obxectos

1.1 Sumario

- 1 Que é un obxecto
- 2 Que é unha clase
- 3 A herdanza
- 4 As interfaces
- 5 Os paquetes
- 6 Exercicios

1.2 Que é un obxecto

Entender o que é un obxecto é esencial para comprender a tecnoloxía de orientación a obxectos na que se sustenta Java. Os obxectos reais están ao noso redor, un can, un coche, unha mesa, etc. Estes obxectos teñen uns atributos (nome, cor, número de rodas, etc.) e poden executar determinadas accións. Por exemplo, un can pode correr, un coche pode aparcarse, etc. Estas accións determinan o comportamento dun obxecto.

Un obxecto software é, conceptualmente, similar a un obxecto real. Ten unha serie de **atributos** que definen o seu estado nun momento dado (variables, nalgunhas linguaxes de programación) e uns **métodos** que determinan o seu comportamento, o que pode facer (funcións, nalgunhas linguaxes de programación). Os métodos permiten cambiar o estado interno dun obxecto, os atributos, e proporcionan un mecanismo de comunicación entre obxectos. Ocultar os atributos dun obxecto, é dicir, o seu estado, e requirir que todas as interaccións dese obxecto se fagan a través de métodos coñécese como **encapsulación dos datos** e é un concepto fundamental da orientación a obxectos.

A orientación a obxectos ten unha serie de vantaxes que a fan especialmente axeitada para proxectos de enxeñería de software, entre elas as seguinte:

- **Modularidade**: Un obxecto ten un código fonte asociado que pode escribirse e manterse independentemente do código fonte doutros obxectos. Unha vez creado e probado un obxecto pode reutilizarse facilmente.
- **Ocultación de información**: Interactuando cun obxecto a través dos seus métodos, os detalles de como está implementado permanecen ocultos ao exterior.
- **Reutilización de código**: Derivado da propiedade de modularidade. Podemos reusar obxectos doutros programadores no noso código.
- **Facilidade de depuración de código**: Se un obxecto concreto dá problemas podemos eliminalo e poñer outro no seu lugar que teña un comportamento axeitado, sen afectar ao resto do sistema, como se fose unha peza dun coche que deixa de funcionar, non tiramos o coche, senón que cambiamos a peza que falla.

1.3 Que é unha clase

Hai que diferenciar entre a clase de obxectos, que describe un grupo de entidades con características comúns, e o obxecto ou instancia, que describe un membro concreto do grupo. Así, unha clase é un prototipo a partir do cal se crean os obxectos, é dicir, é o molde para facer obxectos dun determinado tipo. Aos obxectos chámaselles **instancias** dunha clase. Por exemplo, a clase Bicicleta podería declararse como segue:

```
class Bicicleta {
   // Atributos
   int velocidade = 0;
   int marcha = 1;
   // Métodos
   void cambiarMarcha(int novoValor) {
     marcha = novoValor;
 }
 void acelerar(int incremento) {
   velocidade = velocidade + incremento;
 }
 void frear(int decremento) {
   velocidade = velocidade - decremento;
 }
 void imprimirEstado() {
 System.out.println("Velocidade: "+velocidade+" Marcha: "+marcha);
 }
}
```

A clase Bicicleta ten dous atributos, velocidade e marcha; e tres métodos, acelerar, frear e imprimirEstado.

Un programa en Java, normalmente, estará formado por varias clases. A clase bicicleta define o comportamento dunha bicicleta así como os seus atributos pero, probablemente, formará parte dunha aplicación maior. Cando facemos un programa en Java teremos unha clase especial que é a clase principal que contén un método chamado main. Dende este método main é dende onde se crearán (instanciarán) os diferentes obxectos do noso programa, tal e como vemos no seguinte exemplo, onde se crean varios obxectos de tipo bicicleta:

```
class DemoBicicleta {
   public static void main(String[] args) {
   // Crea dous obxectos bicicleta
   Bicicleta bicicleta1 = new Bicicleta();
   Bicicleta bicicleta2 = new Bicicleta();
   // Invoca os métodos destes obxectos
   bicicleta1.acelerar(10);
   bicicleta1.cambiarMarcha(2);
   bicicleta1.imprimirEstado();
   bicicleta2.acelerar(10);
   bicicleta2.cambiarMarcha(2);
   bicicleta2.acelerar(10);
   bicicleta2.cambiarMarcha(3);
   bicicleta2.imprimirEstado();
 }
}
```

O conxunto de clases que forman un programa e as súas relacións define o modelo co que se representa a realidade no contexto do problema que se ten que tratar.

Cando falamos dun obxecto faise referencia a unha estrutura que existe nun momento determinado da execución do programa, e ao falar de clase, faise referencia a unha estrutura que representa un conxunto de obxectos dunha vez e para sempre.

1.4 A herdanza

Igual que na vida real, existen obxectos que poden compartir características. A herencia é un mecanismo que permite definir unha nova clase a partir dunha anterior mediante a descrición das diferencias entre estas. Por exemplo, unha bicicleta pode ter características específicas se é de carreiras, de montaña, un tándem, etc. Á clase máis xeral, neste caso a bicicleta, chámaselle **superclase** e ás outras **subclases**. As subclases herdan os atributos e os métodos da superclase da que derivan, polo tanto é un mecanismo que permite reutilizar código.

A herdanza é unha ferramenta moi útil porque permite obter uns niveis de reutilización moi elevados, aumentando a produtividade e a fiabilidade das aplicacións, e facilitando tamén o seu mantemento.

En Java unha subclase só pode herdar dunha superclase. A sintaxe para a herdanza é a seguinte:

```
class BicicletaTandem extends Bicicleta {
    // novos atributos e métodos do tándem
}
```

1.5 As interfaces

Podemos especificar o comportamento dunha clase, é dicir, os seus métodos, a través de interfaces. Unha interface dunha clase é un conxunto de métodos públicos pero sen o código, é dicir, é unha declaración pública do que vai facer a clase. O código deses métodos implantase na clase, non na interface. Polo tanto, podemos ver unha interface como un ?contrato? entre a clase e o mundo exterior (as outras clases do programa) mediante o cal unha clase que implementa un interface debe incluír todos os métodos desa interface, xa que se non fose así incumpriría o contrato.

As interface decláranse coa palabra reservada interface, tal e como vemos no seguinte exemplo:

```
interface IBicicleta {
 void cambiarMarcha(int novoValor);
 void acelerar(int incremento);
 void frenar(int decremento);
 void imprimirEstado();
}
```

Para declarar unha clase que implemente a interface anterior teríamos que escribir o seguinte:

```
class Bicicleta implements IBicicleta {
 // Codificación dos métodos públicos
}
```

1.6 Os paquetes

Como xa se comentou, un programa Java pode estar formado por centos de clases. Polo tanto, é necesario dispor dun mecanismo que permita organizalas dalgunha maneira. Este mecanismo son os paquetes que permiten agrupar clases e interfaces relacionadas. A plataforma Java contén moitos paquetes ?de serie?, é o que se chama a API de Java (*Application Programa Interface*).

1.7 Exercicios

1. Visita a dirección http://java.sun.com/javase/6/docs/api/index.html e bótalle un ollo á API de Java.
2. Completa as seguintes frases:

Os obxectos do mundo real conteñen _____ e mais _____. O estado dun obxecto software almacénase en _____. O comportamento dun obxecto software exponse a través de _____. Ocultar os datos internos ao mundo exterior e accedendo a eles unicamente a través de métodos públicos coñecese como _____. O modelo para un obxecto software chámase _____. O comportamento común pode definirse nunha _____ e herdase nunha _____ utilizando a palabra reservada _____. Unha colección de métodos sen ningunha implementación chámase _____. Un conxunto de clases e interfaces organizadas por funcionalidade chámase _____. O termo API quere dicir _____.

1. Crea unha clase que represente un polígono regular que teña como atributos número de lados e lonxitude do lado. Os métodos relevantes serán calcular área, calcular perímetro, establecer número de lados e establecer perímetro. Crea unha clase principal para crear 2 obxectos (1 triángulo equilátero e un cadrado), establece a lonxitude do lado do triángulo a 10, a do cadrado a 15 e imprime o área e o perímetro de ambos. Calcula sempre o área como a lonxitude do lado ao cadrado (independentemente do número de lados que teña o polígono, aínda que esto sexa un erro). Utiliza como modelo a clase Bicicleta.
2. Crea un interface para a clase polígono regular que creaches previamente cos métodos requiridos. Implementa a clase con ese interface previamente definido omitindo algún dos métodos definidos na interface. Que pasa cando compilas?.