INS MANUEL VÁZQUEZ MONTALBÁN

M03. Programació

UF4. Object Oriented Programming

1. Introducció

Let's play!

Feu grups de 3 persones i anoteu les característiques i què es pot fer amb els elements que es mostren a continuació

Quines característiques té i què es pot fer amb un joc de cartes?

Característiques

Accions

5 minuts!

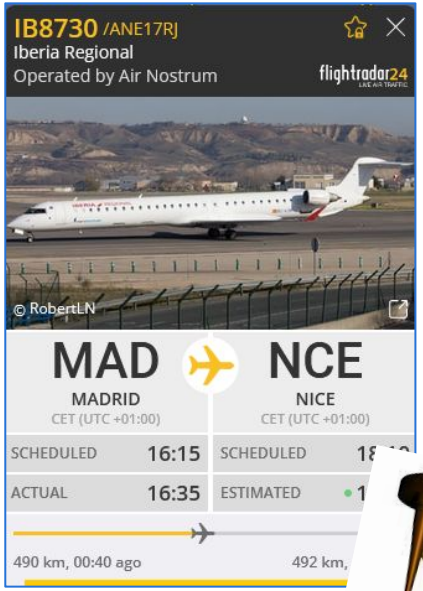
Quines característiques té i què es pot fer amb un llum?

Característiques

Accions

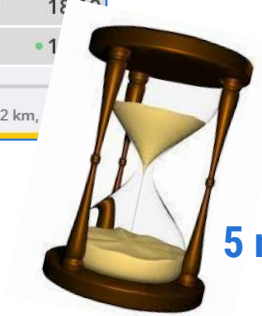
5 minuts!

Quines característiques té i què pot fer un vol?



Característiques

Accions



5 minuts!

Paradigmes de la programació. Orientació a objectes

- **imperatiu**: el programador instrueix a la màquina com canviar el seu estat
  - **procedimental**: agrupa les instruccions en procediments
  - **orientat a objectes**: agrupa les operacions amb la part de l'estat en el qual operen
- **declaratiu**: es declaren les propietats del resultat desitjat, però no com calcular-lo
  - **funcional**: el resultat desitjat es declara com el valor d'una sèrie d'aplicacions de funció
  - **lògic**: el resultat desitjat es declara com la resposta a una pregunta sobre un sistema de fets i regles
  - **matemàtic**: el resultat desitjat es declara com la solució d'un problema d'optimització
  - **reactiu**: es declara el resultat desitjat amb fluxos de dades i la propagació del canvi

Orientació a objectes

En la OOP, tot es treballa dins el concepte d'*objecte*. És a dir, tant les entitats materials (porta, banc, pedra) com els immaterials (compte corrent, hipoteca, factura,...) es defineixen com a objectes. D'aquests, ens interessa definir les seves característiques (edat, material, mida, estat) com els seus comportaments (accions que poden realitzar aquests objectes o que es poden realitzar sobre ells).

Orientació a objectes. Java

Java és un llenguatge d'alt nivell i orientat a objectes que es va llançar l'any 1995 per Sun Microsystems. És un llenguatge de propòsit general, la filosofia del qual resideix en el lema "Write once, run anywhere", ja que gràcies a la seva màquina virtual (JVM) permet executar un codi implementat en Java a qualsevol plataforma, sense haver de tornar-lo a "compilar". Les aplicacions Java es compilen i donen com a resultat un fitxer bytecode. Aquest és el fitxer que la JVM específica de la plataforma s'encarrega d'executar. Aquesta característica, que permet la portabilitat dels programes, juntament amb la gestió dinàmica de la memòria (Garbage Collector) han fet el llenguatge, i tota la plataforma que el suporta, molt popular.

Actualment s'utilitza pel desenvolupament d'aplicacions web (applets, servlets, jsp), aplicacions d'escriptori (SWING i AWT), SGI (CRM, ERPs, …) i d'aplicacions mòbils (android), a més de sectors emergents com ara IA, Big Data (Cloudera, Spark i Scala) i Machine Learning.

Orientació a objectes. Objectes i classes

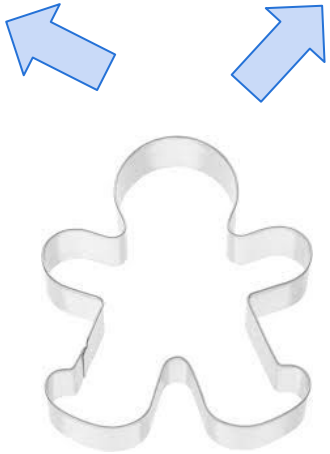
Un **objecte** és una representació (informàtica) d'una entitat del món real (ex: animal, estudiant, cercle, botó,...). Una **classe** és una plantilla (*blueprint*) que defineix les característiques (*atributs*) i els comportaments (*mètodes*) d'un objecte.

És a dir, per a crear (*instanciar*) un objecte, hem de tenir definida en primer lloc la seva classe.

Podem trobar classes ja definides a la API de Java i es poden dissenyar noves classes, específiques per a cada projecte.

Dins la classe, es defineixen els **atributs** (característiques) i els **mètodes** (comportaments) de l'objecte.

Orientació a objectes. Objectes i classes



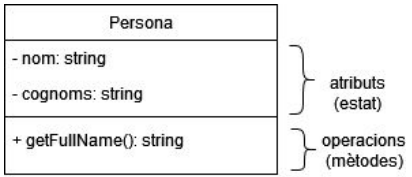
Orientació a objectes. Objectes i classes

Els **atributs** (*instance fields*) són les característiques d'un objecte que defineixen el seu estat, i el seus valors s'emmagatzemen en variables declarades dins la classe, que mantindran valors independents per a cada instància (objecte creat). Els **mètodes** (*comportaments*) són similars a les funcions pròpies dels llenguatges procedimentals i defineixen accions que poden realitzar els objectes o que es poden realitzar sobre ells.

Pseudocodi

```
Classe Persona
    Atributs
        nom, cognoms: string;
    Operacions
        getFullName(): string;
End Persona
```

Diagrama UML



Orientació a objectes. Objectes i classes

Company name: Maserati
Model name: GranTurismo Sport
Fuel type: Premium Gasoline
Mileage: 12 l/100km
Price: 403.000 USD

Procedimental Programming

```
public class Main {
    public static void main(String[] args) {
        String companyName, modelName, fuelType;
        float mileage = 15f;
        double price;

        if(mileage>50){
        }
    }
}
```

Orientació a objectes. Objectes i classes

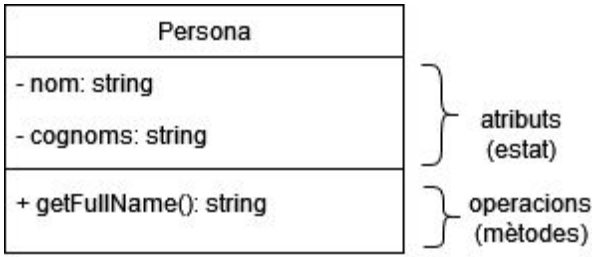
Object Oriented Programming

```java
import cat.institutmvm.utils.Car;

public class Main {
    public static void main(String[] args) {
        Car cotxe1 = new Car("Maserati");
        System.out.println(cotxe1.getCompanyName());

        Car cotxe2 = new Car("Renault");
         System.out.println(cotxe2.getCompanyName());
    }
}
```

```java
public class Car {
    private String companyName, modelName, fuelType;
    private float mileage = 15f;
    private double price;

    public Car(String companyName){
        this.setCompanyName(companyName);
    }

    public String getCompanyName(){
        return this.companyName;
    }

    public void setCompanyName(String companyName){
        this.companyName = companyName;
    }
}
```

Orientació a objectes. Característiques bàsiques

- **State:** represents the data (value) of an object.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.



Orientació a objectes. Característiques bàsiques. Atributs

Els atributs d'un objecte es poden definir mitjançant variables de tipus primitius (comunes amb els llenguatges procedimentals) i/o instàncies d'altres objectes. A més a més, segons definim aquestes variables (modificadors), aquestes poden ser de dos tipus:

- **variables d'instància**: pertanyen a la instància de l'objecte i els seus valors són únics per a aquesta instáncia. Si no existeix la instància, no existeixen ni es pot accedir a aquestes variables
- **variables de classe**: aquests atributs existeixen sempre i es pot accedir als seus valors sense haver d'instanciar un objecte de la classe. Els seus valors són comuns per a totes les instàncies i, si una el modifica, aquest atribut serà modificat per a totes.

La paraula reservada **"this"** permet fer referència a l'objecte actual sobre el qual s'està executant el mètode.

Orientació a objectes. Característiques bàsiques. Mètodes

Cada classe pot tenir un conjunt de mètodes associats amb ella, diversos d'ells amb el mateix nom (*overloading*). El *method signature* (signatura de mètode), diferencia cada mètode, ja que està definit pel nom del mètode i els arguments que rep.

La **signatura del mètode** està formada només per dos dels components de la declaració del mètode: el **nom del mètode** i els tipus i nombre d'arguments.

El tipus de retorn d'un mètode no forma part de la seva signatura, i per tant no serveix per a diferenciar mètodes (en cas d'overloading i/o overriding).

Orientació a objectes. Característiques bàsiques. Mètodes

Classificació dels mètodes:

- **Accessor methods**
  - anomenats habitualment "getter" methods
  - retornen el valor d'una instància específica
- **Mutador methods**
  - anomenats habitualment "setter" methods
  - modifiquen o estableix el valor d'un atribut específic
- **Functional methods**
  - retornen o realitzen algun tipus de funcionalitat (comportament) per l'objecte

Orientació a objectes. Característiques bàsiques. Constructors

Cas particular dels mètodes. D'accés públic i declarats amb el mateix nom que la seva classe, permeten crear una instància d'un objecte.

- els constructors s'invoquen amb la paraula clau *new*
- es pot declarar més d'un constructor en una declaració de classe, sempre i quan tinguin signatures diferents (overloading)
- la crida al constructor ha d'incloure arguments que coincideixin amb la llista d'arguments del constructor
- si no es declara un constructor, Java proporcionarà un constructor per defecte (blank)
- quan es declara un constructor propi, el constructor per defecte proporcionat per Java ja no estarà disponible
- els constructors assignen valors inicials a les variables d'instància d'una classe
- els constructors dins d'una classe es declaren com a mètodes però no retornen cap valor

Orientació a objectes. Encapsulament

Ocultació de les decisions de disseny, de manera que canvis en una classe afectin el mínim possible el programari ja desenvolupat. Els ***access modifiers*** (modificadors d'accés) especifiquen l'accessibilitat per a modificar variables, mètodes i classe.

Hi ha quatre tipus de modificadors d'accés:

- **public**: l'element és accessible per a tothom. El menys restrictiu.
- **protected**: l'element és accessible només si la classe accessora (accessing class) està al mateix package o si la classe accessora és una subclasse dins de qualsevol altre package.
- **private**: només és accessible des de dins de la mateixa classe. És el més restrictiu.
- **default** (blank): també conegut com "package private", s'aplica quan no s'indica el modificador d'accés. Només és accessible per classes i subclasses del mateix package (default).

Orientació a objectes. Encapsulament

Els **access modifiers** especifiquen l'accessibilitat per canviar variables, mètodes i classes. Els camps són normalment privats i els mètodes públics. En cas de declarar una variable o mètode sense cap modificador de control d'accés, estant disponibles per a altra classe dins del mateix paquet.

- Qualsevol variable declarada sense un modificador pot ser llegida o modificada per una altra classe del mateix paquet.
- Qualsevol mètode declarat sense un modificador pot ser invocat per qualsevol classe en el mateix paquet.

Un **objecte** ha de protegir sempre el seu **estat** (atributs), per tal de no permetre accessos directes. Els **atributs** han de ser declarats de manera **privada** i **accessibles a través dels mètodes**.

Orientació a objectes. Encapsulament

Els **non-access modifiers** (modificadors de no accés), no canvien l'accessibilitat de variables i mètodes, sinó que poden alterar el comportament dels mètodes i els hi proporcionen propietats especials:

- **static**: en el cas dels atributs i els mètodes, fa que siguin dependents de la classe (no de la instància). En el cas de les classes, encapsulen funcionalitats típiques a utilitzar en les nostres aplicacions
- **final**: aplicable a les classes, mètodes i variables per a indicar que no es canviaran
- **abstract**: aplicable a classes i mètodes, els defineix com a abstractes (declarats però no definits)
- **syncronized**: utilitzat en la sincroniització de threads (fils)