Objec Oriented Design

Master 1 of Informatics I.E.I. - V.N.U.

2020-2021

Marie Beurton-Aimar

Université de Bordeaux

Compilation

How to produce a program

- Write a source file - respecting the java language rules.
- For example write a class `Person` and save in a file `Person.java`.
- Compile to produce byte code : in a terminal window write `javac Person.java`.
- If no compiler error, you have now a new file with the name `Person.class`

Compilation

How to produce a program

- Write a source file - respecting the java language rules.
- For example write a class `Person` and save in a file `Person.java`.
- Compile to produce byte code : in a terminal window write `javac Person.java`.
- If no compiler error, you have now a new file with the name `Person.class`
- Verify ! if not verify your class name.
- To run the program : in a terminal window write `java Person`

Compilation

How to produce a program

- Write a source file - respecting the java language rules.
- For example write a class `Person` and save in a file `Person.java`.
- Compile to produce byte code : in a terminal window write `javac Person.java`.
- If no compiler error, you have now a new file with the name `Person.class`
- Verify ! if not verify your class name.
- To run the program : in a terminal window write `java Person`
- In this case, the `main` function has to be written in the `Person.java` file.

The code

```
public class Person{
    private String name;
    private int age ;
    public Person(String chain, int val){
      name=chain;
      age = val;
    }
    public void display(){
      System.out.println("this person is " + name);
      System.out.println
                    ("he/she is " + age + "year old");
    }
    public static void main (String []arg){
      Person item1=new Person(``Lila'',2);
      Person item2 = new Person(``Tung'',4);
      item1.display();
      item2.display();
    }
```

Polymorphism

Methods overloading

- Same method name with different implementations.
- Signatures have to be different:

```
public void addAge()
{ age++; }
pubic void addAge (int val)
{ age +=val; }
```

Compile with multiple files

Separate the `main`

- Create a new `Main.java` file and put into the main.
- Suppress the `main` function in the `Person.java` file.

```
public class Main{
    public static void main (String []arg){
      Person item1=new Person(``Lila'',2);
      Person item2 = new Person(``Tung'',4);
      item1.display();
      item2.display();
    }
}
```

Compile with multiple files

Compile the `main`

- Write in the terminal window:
  `javac Main.java`
- Observe that now you have a `Main.class`
- Run the program by executing the `Main` :
  `java Main`
- The `Main` will be able to use the code in the other file because they are in the same directory(folder).

If you use an IDE some steps of the process could be automatized but remember these rules because you have to respect them to work well.

Attributes of Class object

Keyword `static`

- Attributes/Variables declared in the class are by default attributes of each object - see `name` for example.
- It is possible to declare attribute of class with the keyword `static`

```
public class Person{
    private String name;
    private int age ;
    public static int num=0;

    public Person(String chain, int val){
        name=chain;
        age = val;
        num++;
    }
```

Attributes of Class object

```
  public class Main{
    public static void main (String []arg){
        Person []myArray= new Person[10];
        for (i=0;i<10;i++){
          System.out.println("give a name");
          String aName=getString();
          System.out.println("give an age");
          int aVal=getInt();
          Person item=new Person(aName,aVal);
           myArray[i]=item;
}
        System.out.println
           ("you have " + Person.num + "person in
                                        your array");
```

Attributes of Class object

Keyword `static`

- Notice : access to the attribute is given from the class name not from a variable name.
- Example of `System.out` !

Static method

- A method can be also `static` : `Math.sqrt()`
- No need to create a variable to access to this method.
- Remember `main` is `static`
- Why ?

Attributes of Class object

Keyword `static`

- Notice : access to the attribute is given from the class name not from a variable name.
- Example of `System.out` !

Static method

- A method can be also `static` : `Math.sqrt()`
- No need to create a variable to access to this method.
- Remember `main` is `static`
- Why ?
- `java Main` calls the method Main.main().
- Consequence: methods called by a `static` method have to be `static` too.

Final keyword

- `final` keyword allows to specify that a variable cannot be modified, i.e. it is a constant.
- The initial value is given at the declaration step.
  `public final int MAX = 10;`
- A `final` method cannot be overload by another class.
- A `final` method is not inherited by subclass.
- Allows to make more secure the application.

Inheritance

- Object oriented design allows to specify concepts (or functionalities) that are shared by several classes (types).
- In this case, we define a generic (super) class for that and particularities are given in subclasses which inherit of the super class.
- A subclass inherits of all variables and methods : public and protected, belonging to the super class.

A part !

public, private and protected keywords

- Both attributes and methods of an object are qualified by 3 possible keywords that define their *visibility*:
  - `public`: available from everywhere - if the `classpath` is correctly set. Value by default if no precision.
  - `private`: available only inside the code of the class, hidden for all other classs objects. Compiler errors if it is not respected.
  - `protected`: available in the class and in all its subclasses if any. Compiler errors if not respected
- Defining `package` modifies the default rules of visibility. In a first attempt we do not create any package. In this case, a default package is created with the current folder.

Inheritance

- Key-word `extends` allows to declare inheritance of a class.

Genericity

- Behaviors (methods) of objects are ready without knowledge of the used object type(class).
- An object can react to message without knowledge of the sender type (client).

Keyword `super`

- Calling the constructor of the super class by `super()` with or without parameters depending on the case.
- This calling has to be the first instruction in the constructor of the subclass.

Java Language

```
public class Student extends Person{
  private String diploma;
  public Student(String chain, int val,
                 String level)
  {
    super(chain, val);
    diploma = level;
  }
  public void display()
  {
   super.display();
   System.out.println("Diploma is " + diploma);
  }
}
```

Abstract Class

Goal

- To give a not complete implementation of a class.
- To forbid instanciation.

How to

- Declare the class as abstract.
- Use the keyword `abstract` :
  `public abstract MyAbstractClass`
- Has a meaning only if the abstract class is a super class.
- A lot of example in the java library :
  `AbstractList, AbstractQueue, AbstractSet`
  `....`
- A method can be `abstract` in this case it is mandatory to declare the class as `abstract` too.

Abstract Class

Example

- You have two subclasses of players : tennis and golf.
- You have a super class `Player` and the two subclasses inherits from `Player`.
- The class `Player` has a method `play()` but as it differs in the two subclasses we don't want to give a generic code in the super class.
- We declare in the class `Player` the method `public abstract void play()` and we give the code of the method in the subclasses.
- Consequence: `Player` class is abstract too.

Your turn!

Exercise

- Write a class `Person` with a name and an age.
- Write a subclass `Player` with an array of 10 scores (integer).
- Write a class `Game` which contains the `main` function, that create 2 players, ask to the user the scores they have obtained .
- Display the scores of each players and claim who is the winner, i.e with the best sum of scores.