Software

Lab session 1

OO design & programming I

- Object = instance of a class, working entity
- Class = model of what an object can do
- Method = can modify the state of a class => what a class can do, behaviour
- Fields, properties: state of a class = data.

OO design & programming II: 4 basic principles

- 1. Encapsulation

Encapsulation is accomplished when each object maintains a private state, inside a class. Other objects can not access this state directly, instead, they can only invoke a list of public functions. The object manages its own state via these functions and no other class can alter it unless explicitly allowed. In order to communicate with the object, you will need to utilize the methods provided. (getters & setters)

OO design & programming II: 4 basic principles

2. Abstraction

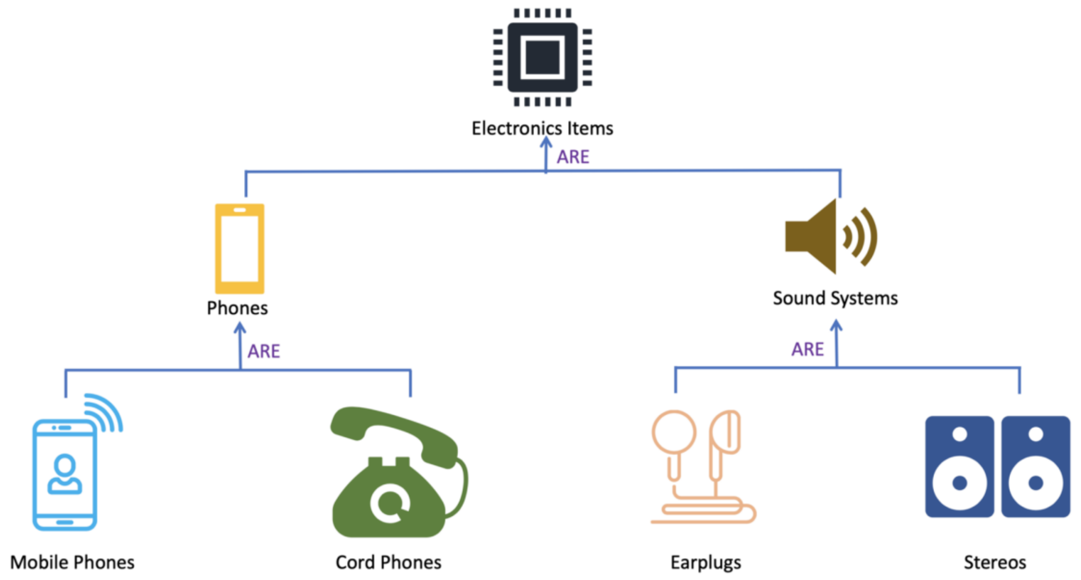
Abstraction is an extension of encapsulation. It is the process of selecting data from a larger pool to show only the relevant details to the object.

=> Omit all details that are not relevant

OO design & programming II: 4 basic principles

3. Inheritance

Inheritance is the ability of one object to acquire some/all properties of another object. For example, a child inherits the traits of his/her parents. With inheritance, reusability is a major advantage. You can reuse the fields and methods of the existing class.



Inheritance II

- Inheritance does not solve all problems
  - Best practice: limit to max 3 levels; otherwise very surprising and uncomfortable side-effect WILL show up. => maintenance and testing becomes very tedious.

  - One size does NOT fit all. In many cases methods will have a different implementation in the superclass and in the specialized class. To solve that problem: the child class can OVERRIDE the method in the base class
    - How? Just redefine the method in the child class
  - Visibility :
    - Private: only accessible from within the class
    - Protected: accessible from within the class + from child classes
    - Public : accessible from all classes.

OO design & programming II: 4 basic principles

4. Polymorphism

gives us a way to use a class exactly like its parent so there is no confusion with mixing types. This being said, each child sub-class keeps its own functions/methods as they are. If we had a superclass called Mammal that has a method called mammalSound(). The sub-classes of Mammals could be Dogs, whales, elephants, and horses. Each of these would have their own iteration of a mammal sound (dog-barks, whale-clicks).

- Example: Animalshelter
- Cats & Dogs are both pets.
- Cats kill mice, dogs bark
- The shelter keeps pets.
- They need to walk with dogs
- They need to feed the animals
- …

Learn IntelliJ

- Tutorials on the Internet
- IntelliJ : Help -> Learn IDE features
- Reference, how to’s :
  - https://www.jetbrains.com/help/idea/getting-started.html

- Most important: many many hours of practice.

GIT (very short intro)

- = Version control system
- = database of all versions of all files in a project
- This database is called REPOSITORY (in server, in cloud eg Github)
- Contributors push a set of files to the repository. This event is called a COMMIT
- Download and install from https://git-scm.com/
- GIT is built in in Visual Studio, IntelliJ, => most popular VCS
- CLI : rather difficult to master
- Graphical tools: SourceTree, Git Extensions, .. (https://git-scm.com/downloads/guis)

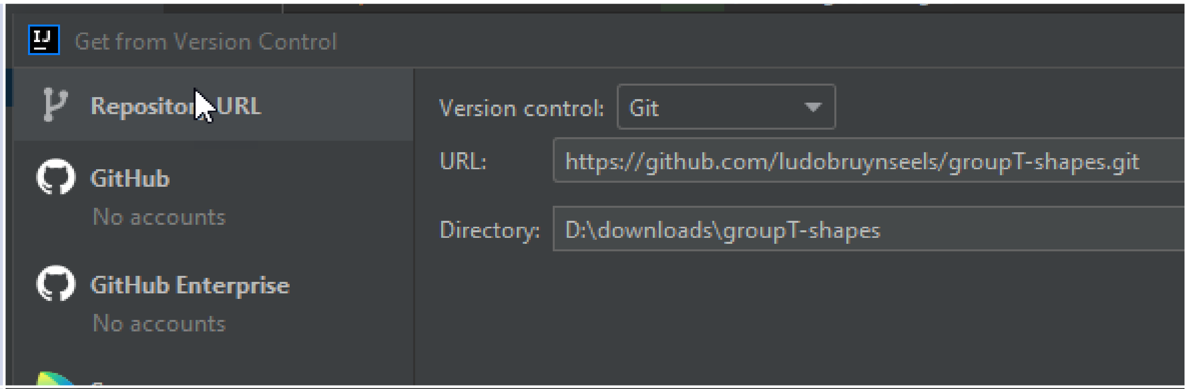
Assignment 3: Refactor ‘Shapes’

Refactor ‘Shapes’

- **Launch IntelliJ**
- **File – New from VCS**

Copy and paste (ctrl + v) this URL:

https://github.com/ludobruynseels/groupT-shapes.git

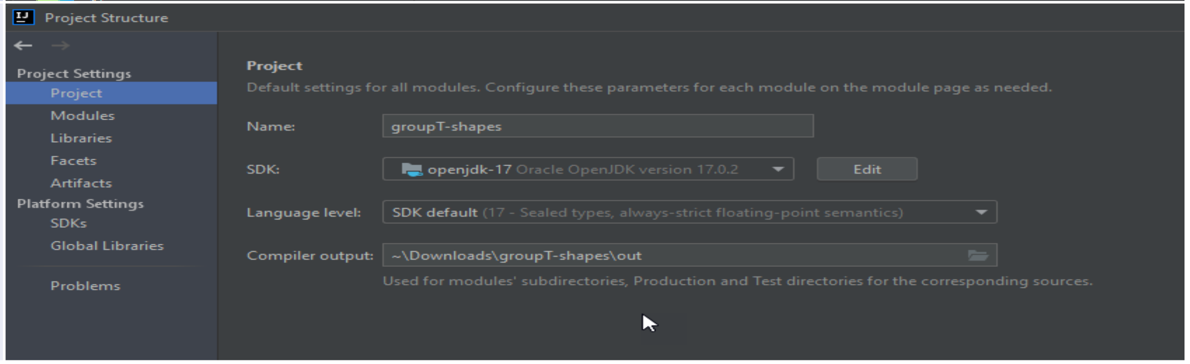


Version control: Git

URL: https://github.com/ludobruynseels/groupT-shapes.git

Directory: D:\downloads\groupT-shapes

File -> Project Structure : Make sure a JDK is selected!



Project

Default settings for all modules. Configure these parameters for each module on the module page as needed.

Name: groupT-shapes

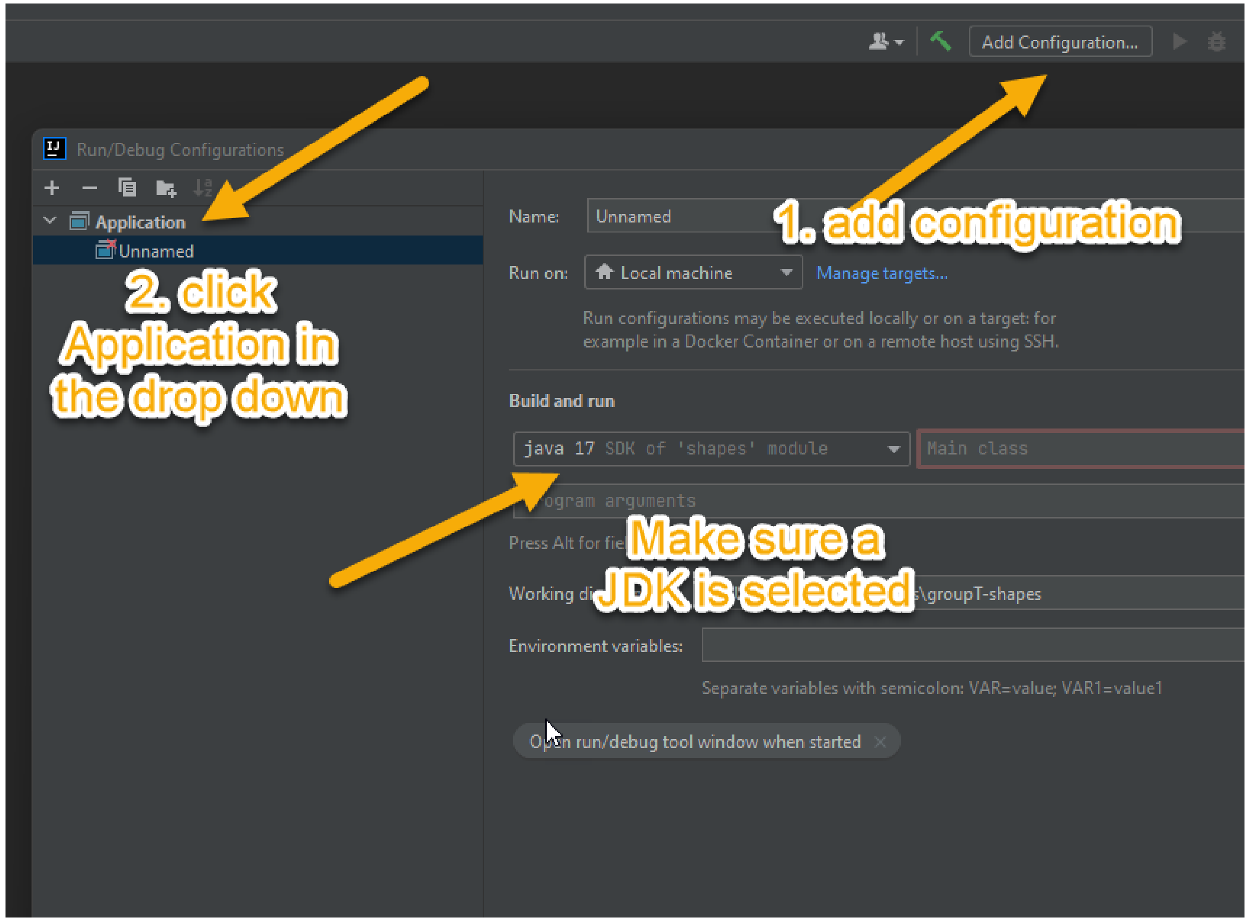
SDK: openjdk-17 Oracle OpenJDK version 17.0.2

Language level: SDK default (17 - Sealed types, always-strict floating-point semantics)

Compiler output: ~\Downloads\groupT-shapes\out

Used for modules' subdirectories, Production and Test directories for the corresponding sources.

Configure your project



Name: Unnamed

Run on: Local machine Manage targets...

Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Build and run

java 17 SDK of 'shapes' module Main class

Program arguments

Press Alt for fie

Working di \groupT-shapes

Environment variables:

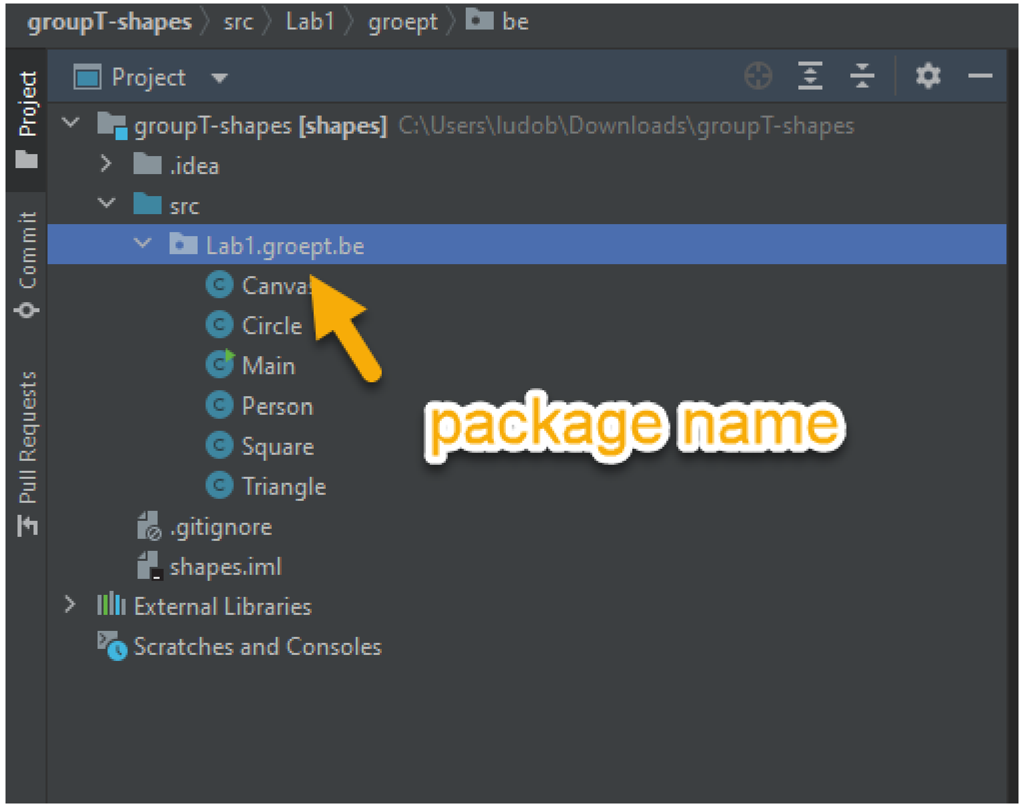
Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

Refactoring

- Design a parent class ex MyShape
- Move as much code from the basic shapes to MyShape
- Subsequently add a class SimpleDrawingProgram, containing a polymorphic collection of MyShape objects.
- Implement the functionality to add a new MyShape object and to draw all of the shapes in the collection. You can test this class by attempting to draw your car from the first lab of the course of Object-oriented Programming and Databases.
- When this works correctly, add a new shape: a rectangle (hint: check the class Square). Test the functionality of this new class by adding a few rectangles of different colors to your drawing.

Packages and fully qualified names



Lib1.groept.be.Canvas

is not the same as

java.awt.Canvas

```
package Lab1.groept.be;

import javax.swing.*;
import java.awt.*;
import java.util.List;
import java.util.*;

public class Canvas
{…}
```