14013103-4

ADVANCED PROGRAMMING

LECTURE 1

Created by Randah Alharbi & Bushra Algotiml

Outline

- Programming Motivation
- The differences between Procedural and Object-oriented Languages
- Object-Oriented Programming (OOP)
- OOP Advantages
  - Abstraction and Encapsulation
  - Exceptions handling
  - OOP libraries
  - OOP in large scale projects

Programming Motivation

- Imagine the life without programming?
  - Supermarket
  - Mobile phone
  - Factories
  - Universities.
  - Space
  - Weather Predictions
  - Hospital
- How can you help the world to be better world for others by programming.

Procedural languages Vs. Object-Oriented languages

- Programming languages allow programmers to code software.
- The three major families of languages are: machine languages, assembly languages and high-level languages
- Historically, we divide High-level languages into two groups:
  - Procedural languages
  - Object-Oriented languages (OOP)

1- Procedural languages:

- Early high-level languages are typically called procedural languages.
- Procedural languages are characterized by sequential sets of linear commands. The focus of such languages is on structure.
- Examples include C, COBOL, Fortran, LISP, Perl, HTML, VBScript

Procedural languages Vs. Object-Oriented languages

2- Object-Oriented languages:

- Most object-oriented languages are high-level languages.
- The focus of OOP languages is not on structure, but on modeling data.
- Programmers code using “blueprints” of data models called classes.
- Examples of OOP languages include C++, Visual Basic.NET, Python and Java.

Procedural languages: Example

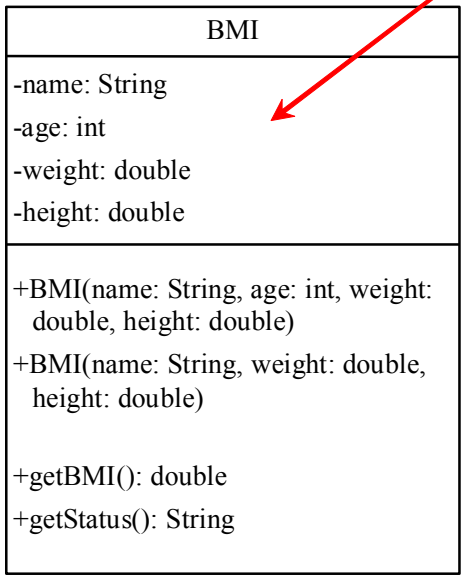
- Lets assume that we have lines of code inside main method for computing body mass index. The code cannot be reused in other programs, because the code is in the main method
- To make it reusable define a static method to compute body mass index as follows:

```
public static double getBMI(double weight, double height)
```

- This method is useful for computing body mass index for a specified weight and height.
- Suppose you need to associate the weight and height with a person's name and birth date. You could declare separate variables to store these values, but these values would not be tightly coupled.

Object-Oriented languages: Example

- The ideal way to couple them is to create an object that contains them all. Since these values are tied to individual objects, they should be stored in instance data fields. You can define a class named BMI



The get methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The name of the person.

The age of the person.

The weight of the person in pounds.

The height of the person in inches.

Creates a BMI object with the specified name, age, weight, and height.

Creates a BMI object with the specified name, weight, height, and a default age 20.

Returns the BMI

Returns the BMI status (e.g., normal, overweight, etc.)

Object-Oriented Programming (OOP)

- Software design using the object-oriented paradigm focuses on objects and operations on objects.
- OOP simplifies software development and maintenance by providing some concepts:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

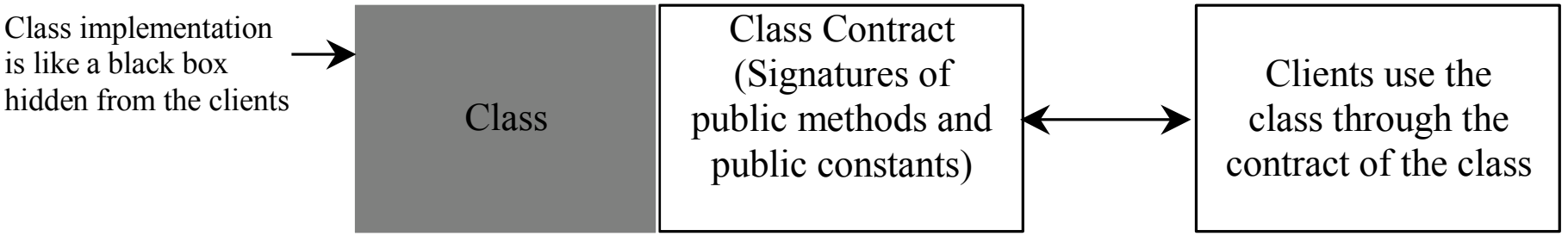
OOP Advantages

- Abstraction and Encapsulation
- Exception handling
- OOP libraries
- OOP in large scale projects

Abstraction and Encapsulation

- Java provides many levels of abstraction, and class abstraction separates class implementation from how the class is used.
- The creator of a class describes the functions of the class and lets the user know how the class can be used. The collection of methods and fields that are accessible from outside the class, together with the description of how these members are expected to behave, serves as the class's contract.
- The user of the class does not need to know how the class is implemented. The details of implementation are encapsulated and hidden from the user. This is called class encapsulation.
- For example, you can create a Circle object and find the area of the circle without knowing how the area is computed. For this reason, a class is also known as an abstract data type (ADT).

Abstraction and Encapsulation



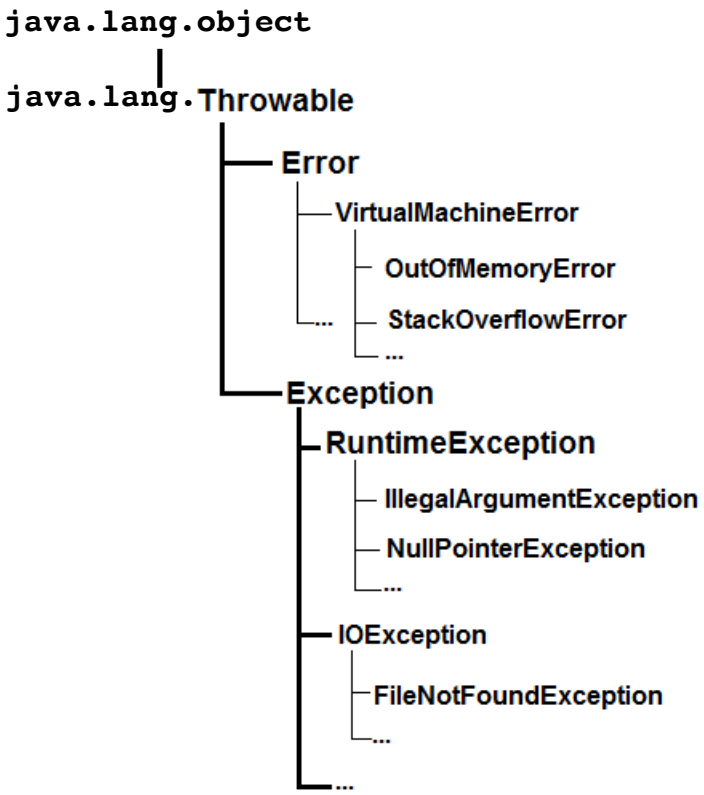
- Class abstraction is the separation of class implementation from the use of a class.
- The details of implementation are encapsulated and hidden from the user. This is known as class encapsulation.

Exceptions Handling

- In Object-Oriented Programming (OOP), exceptions are a powerful mechanism for centralized processing of errors and exceptional situations. This mechanism replaces the procedure-oriented method of error handling in which each function returns a code indicating an error or a successful execution.
- Java's exception handling mechanism is a clean, well-lighted way to handle exceptional situations that pop up at runtime.
  - A method can throw an exception when something fails at runtime.
  - An exception is always an object of type Exception.
  - A method throws an exception with the keywords **throw**, followed by a new exception object.

Exceptions Handling

Java Exception API Hierarchy



Reference: https://www.codejava.net/java-core/exception/java-checked-and-unchecked-exceptions

Exceptions Handling

Types of Exceptions:

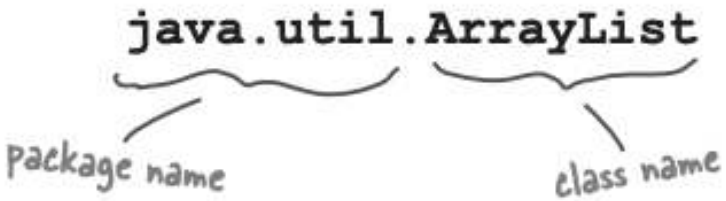
1- Built-in Exceptions such as:

1. `ArithmeticException`
2. `ArrayIndexOutOfBoundsException`
3. `ClassNotFoundException`
4. `FileNotFoundException`
   `IOException`
5. `InterruptedException`
6. `NoSuchFieldException`
7. `StringIndexOutOfBoundsException`

2- User-defined Exceptions: Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called ‘user-defined Exceptions’.

OOP libraries

- Java ships with hundreds of pre-built classes.
- You don’t have to reinvent the wheel if you know how to find what you need from the Java libraries.
- Java libraries are collected in what commonly called JAVA API.
- In the Java API. Classes are grouped into packages.
- To use a class in the API, you have to know which package the class is in.
- You have to know the full name of the class you want to use in your code.
- Example:



OOP libraries

- During the course, you will use different types of JAVA API libraries in your coding.
- Examples:
  - For GUI: `JavaFX`
  - For Event-driven: `JavaFX`
  - For Generics Collections: `util.Collection`
  - For I/O: `io.InputStram, io.OutputStream`

OOP in large scale project

- Object Oriented Design Techniques are widely accepted due to:
  - Simplicity due to abstraction.
  - Easily decomposed into subproblems (easy for teamwork).
  - Better understandability.
  - Easily maintainable.
  - Reuse of Code and Design.
  - Improvement in the productivity

References:

- Intro to Java Programming, Comprehensive Version, 11th Edition, Y. Daniel Liang, Pearson.
- Head First Java, 2nd Edition, Kathy Sierra, O'Reilly
- Dr. Atif Naseer Slides for Advance Programming Course.