1DV532 – Starting Out with Java

Lesson 6

Object-Oriented Programming – Basic Concepts and Principles

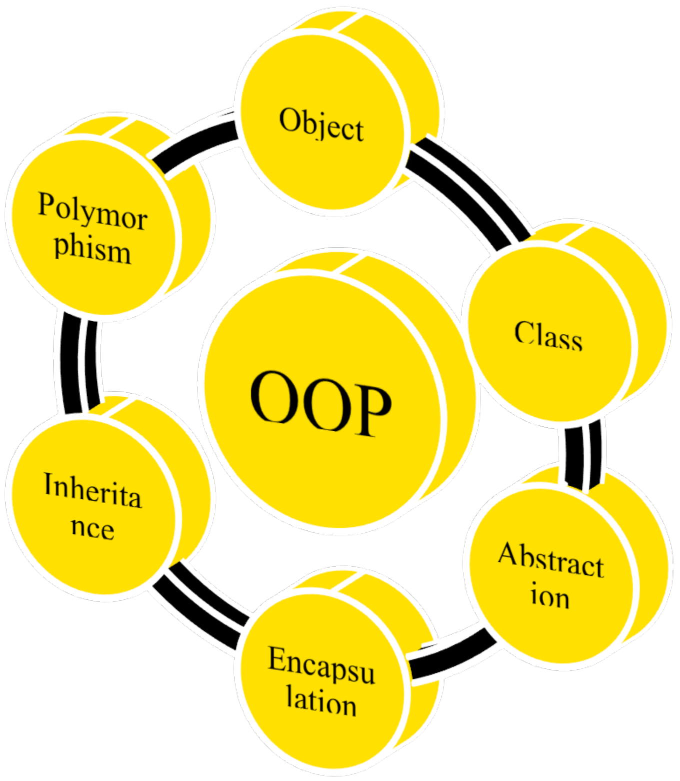
Dr. Nadeem Abbas
nadeem.abbas@lnu.se

Linnæus University

Object-Oriented Programming (OOP)

- OOP is a programming paradigm based on the concept of **Objects** to model and implement solutions for real-world problems

- An object-oriented system (program) is a set of interacting objects.

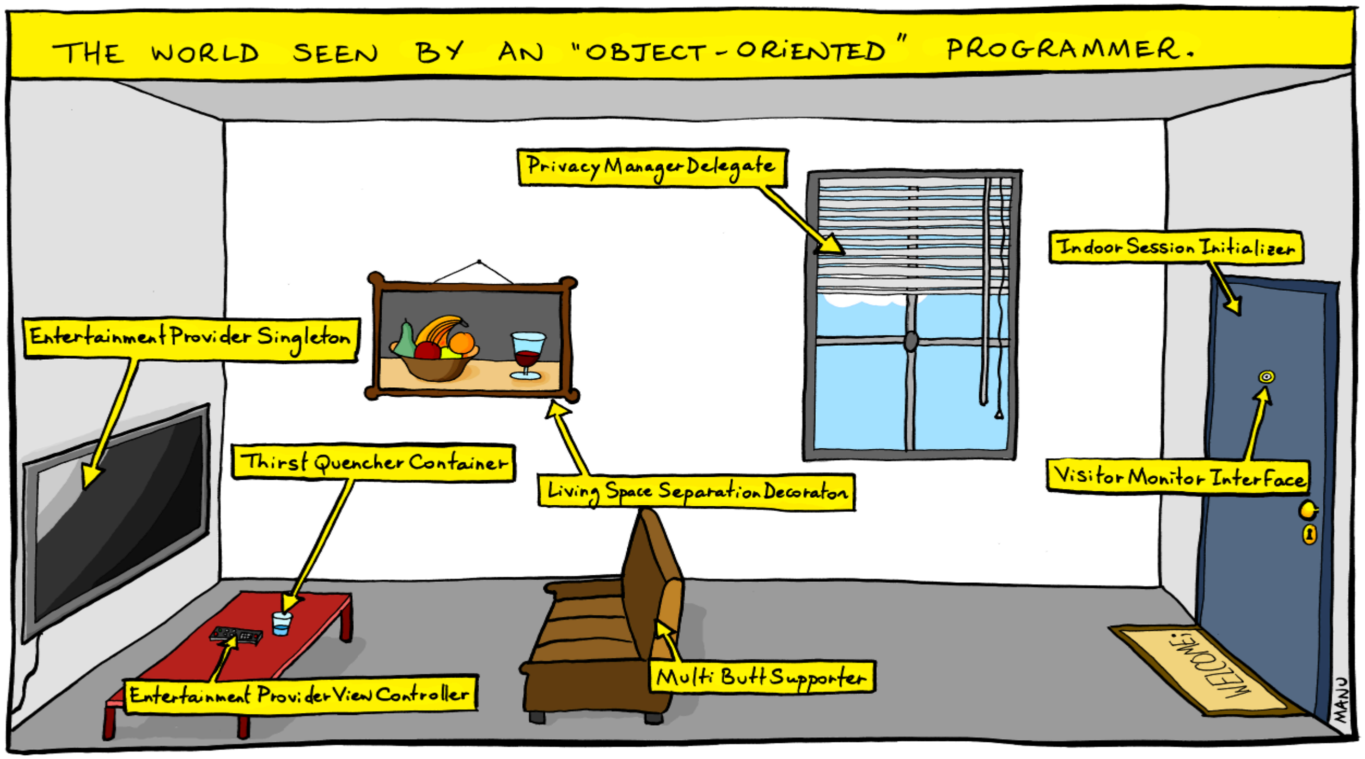


Object-Oriented Programming (OOP)

- Our world is built of objects
  - By looking around, one may find a number of objects
  - While reading this content, you are likely to be seated on some object, most likely a chair, sofa, couch, or something similar.

Object-Oriented Programming (OOP)

- In step 1, we learned that basic objective of programming is *problem solving*
  - The problems belong to the world around us and involve number of *objects*
- *Object-Oriented Programming* provides a natural solution to *visualize, model and develop* solutions for real world problems in terms of *objects*.


THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.
Privacy Manager Delegate
Indoor Session Initializer
Entertainment Provider Singleton
Thirst Quencher Container
Living Space Separation Decorator
Visitor Monitor Interface
Multi Butt Supporter
Entertainment Provider View Controller
WELCOME!
MANU

What is an Object?

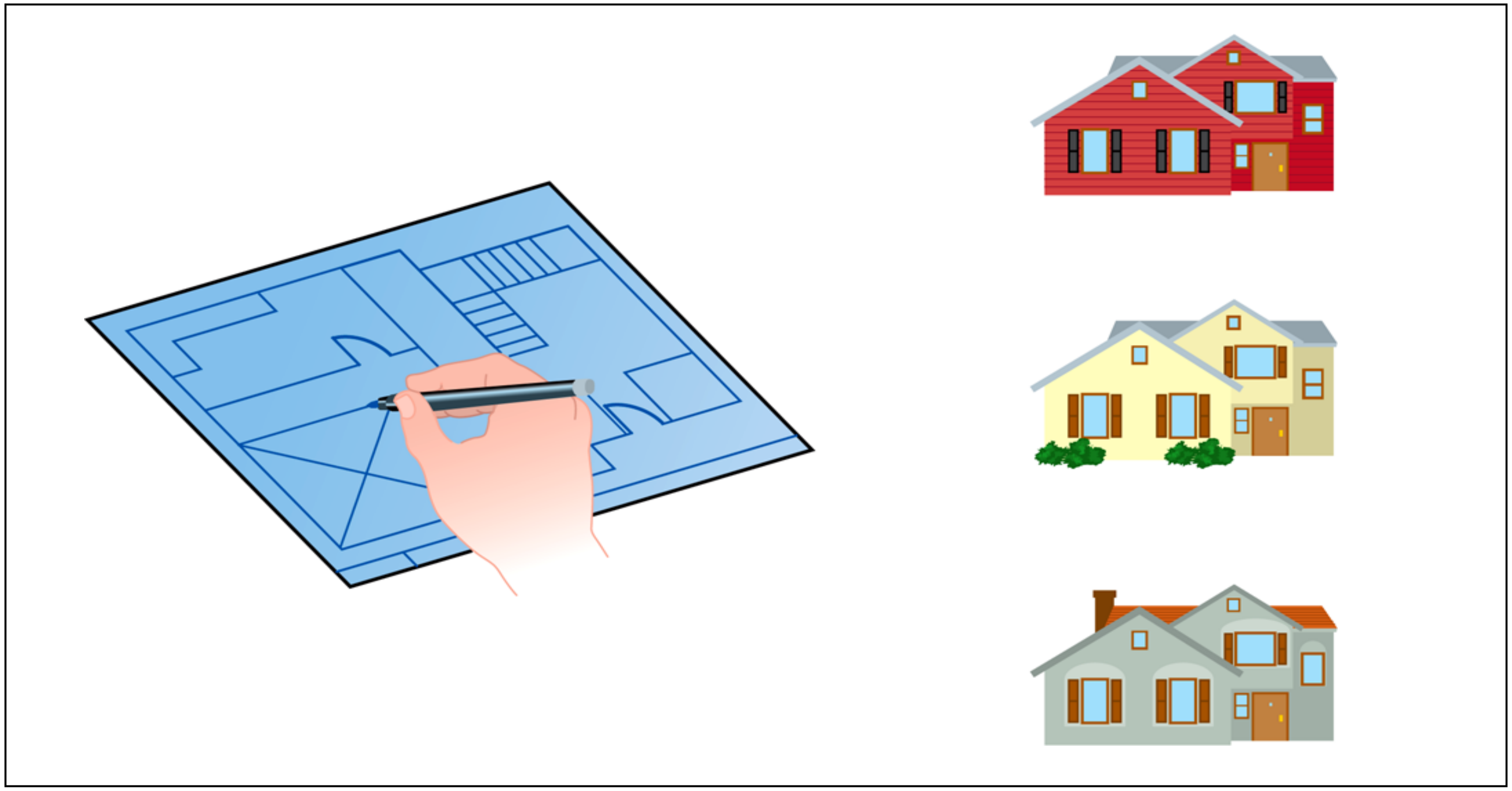
- A real world object is something which is
  - Tangible or physical, e.g., a car, a tree, a house
  - Intangible or conceptual that can be comprehended intellectually, e.g. salary, date, time,

What is an Object?

- Each object has
  - **state (attributes or fields),**
  - **behavior (operations or methods),**
- **State** of an object is a set of fundamental attributes that define or identify an object
  - State of a Person object is defined by attributes, such as, *name, weight, height, address*, etc.
- **Behavior** of an object are the operations an object can perform.
  - speak, walk, run, eat, sleep, etc. are the operations that define behavior of the Person object
- **What are the states and behaviors of a car object?**
- From OOP perspective, **objects** are abstractions with an interface of named **operations** and a hidden local **state**, and have an associated **type (class)**
  - Each object is an instance of a certain **Class**

What is a class?

- A class is a blueprint or prototype from which objects are created.



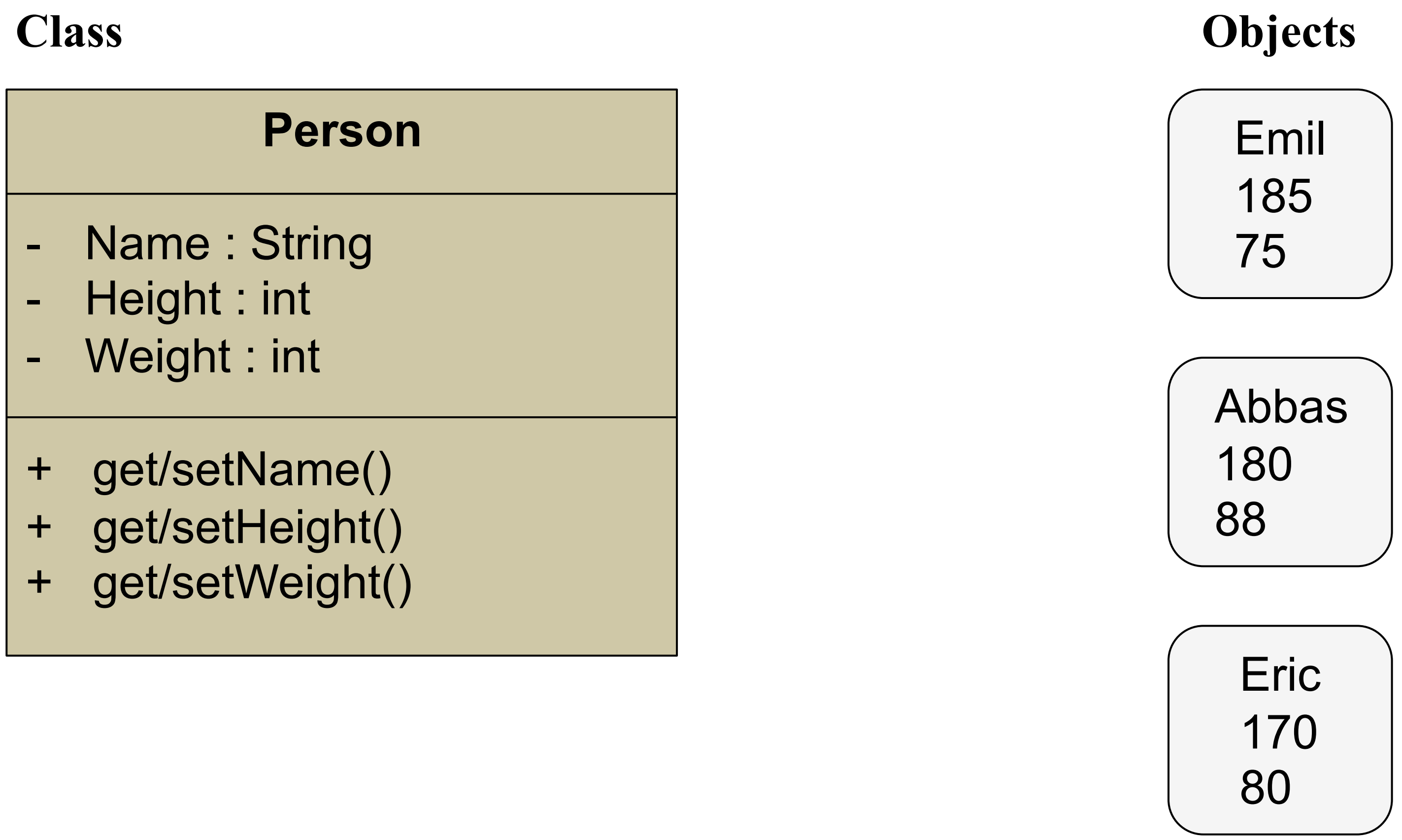
What is a class?

- A class represents a set of object that share a common set of states and behavior (implemented by fields and methods)

Class Example 1: Time	
//Attributes	//Behavior
int hours	SetHours(int h)
int minutes	SetMinutes(int m)
int seconds	SetSeconds(int

Class Example 2: Person	
//Attributes	//Behavior
string name	SetName(string n)
int length	SetLength(int l)
int weight	SetWeight(int w)

Example – Person Class – UML Diagram



Java Class Definition – General Form

```
[access modifier] class ClassName {
      [access modifier] [static] type variable1;
      [access modifier] [static] type variable2;
      ...
      [access modifier] [static] type variableN;

      //constructors
      [access modifier] ClassName (parameter-list) {
                //body of constructor
      }
      [access modifier] [static] type methodname1(parameter-list) {
      // body of method
      }
      [access modifier] [static] type methodname2(parameter-list) {
      // body of method
      }
      // ...
      [access modifier] [static] type methodnameN(parameter-list) {
      // body of method
      }
}
```

Details about the Java Class Definition are discussed in next lesson.

Java Class Definition – Example (Person.java)

```
[access modifier] class ClassName {
[access modifier] type instance-variable1;
[access modifier] type instance-variable2;
...
[access modifier] type instance-variableN;

[access modifier] type
    methodname1(parameter-list) {
    // body of method
}
[access modifier] type
    methodname2(parameter-list) {
// body of method
}
// ...
[access modifier] type
    methodnameN(parameter-list) {
// body of method
}
}
```

```
public class Person {
    /* fields */
    private String name;
    private int height;
    private int weight;
    private static int personCounter;

    /* Constructors */
    public Person(String n, int h, int w) {name = n;
    height = h;weight = w; personCounter++;}
    public Person() {name = ""; height = 0;weight = 0;
    personCounter++;}

    /* methods */
    public void setName(String n) {name = n;}
    public void setHeight(int h) { height = h;}
    public void setWeight(int w) { weight = w;}
    public static int getPersonCounter() {return
    Person.personCounter;}

    public void printPerson() {
    System.out.println("Name: " + name + ", Height: " +
    height + ", Weight: " + weight);
    } }
```

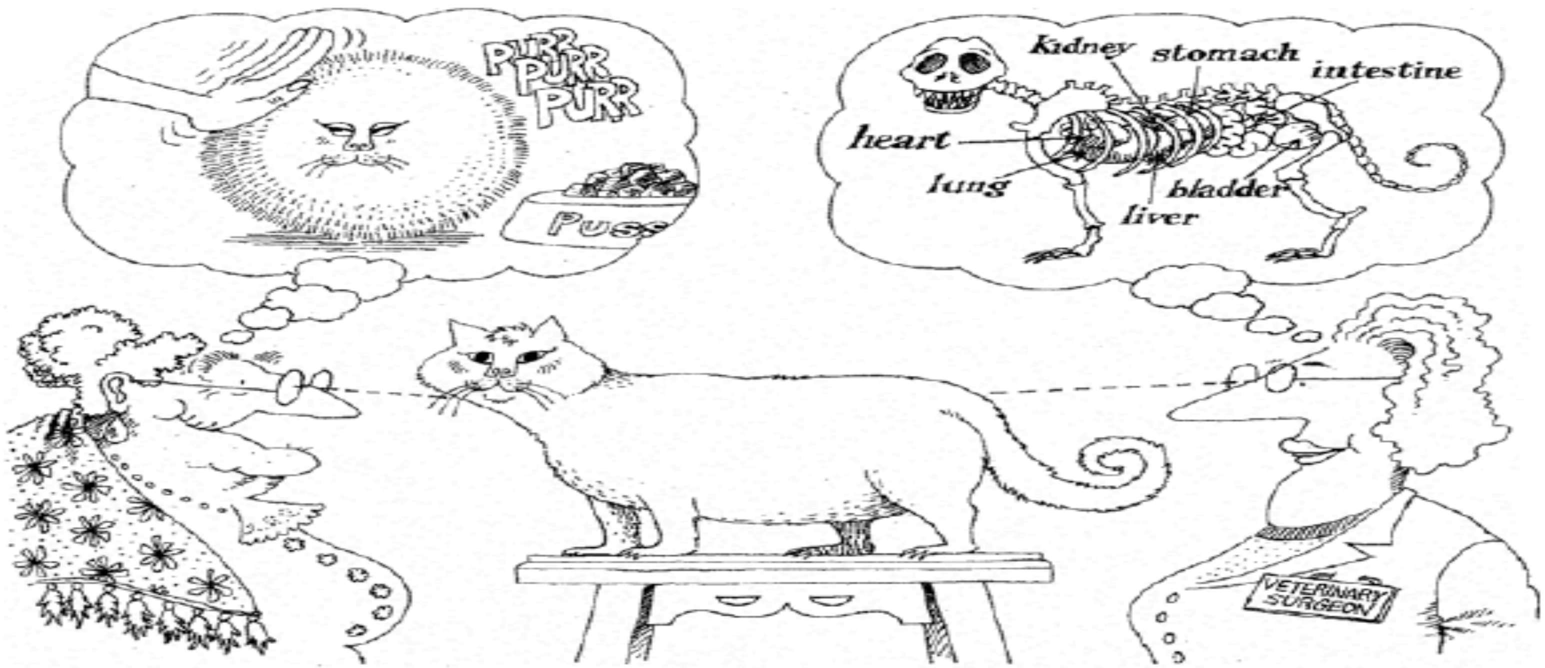
Object-Oriented Programming – Key Concepts

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Abstraction

Basic Idea: "*Capture only those details about an object that are relevant to current perspective*"

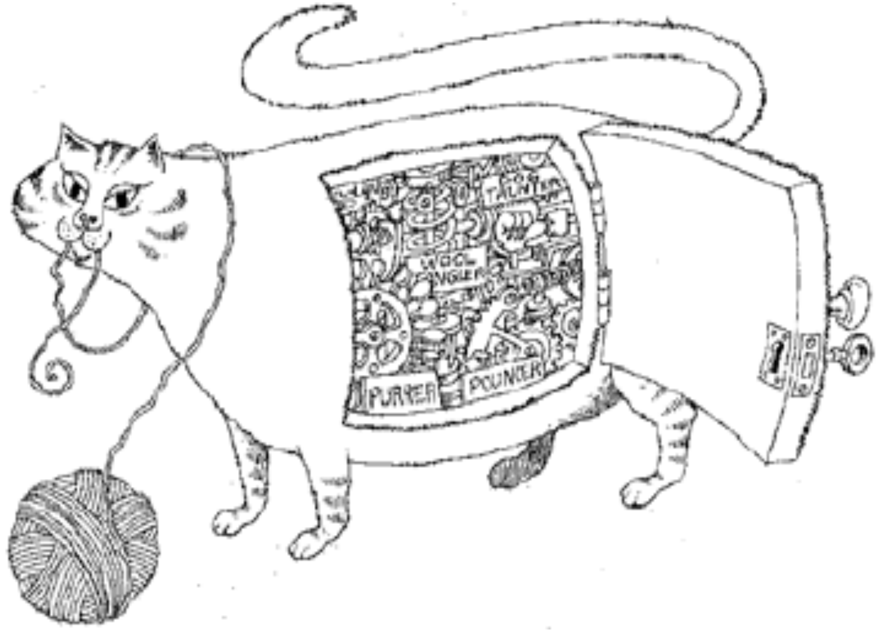
- Abstraction is a fundamental way to mitigate complexity and simplify things (objects) both in real world and in software development
- Abstraction is achieved by focusing on essential characteristics of an object from a particular viewer's perspective.
  - Cat's ordinary perspective: a pet with friendly behavior, four legs to jump around
  - Cat's Surgeon perspective: a patient who needs care for its internal body parts.



Encapsulation – Information Hiding

Basic Idea: *"Show only those details to the outside world which are necessary for the outside world and hide all other details"*

- Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.
  - It serves as a protective shield that prevents data from being accessed by the code outside the shield.



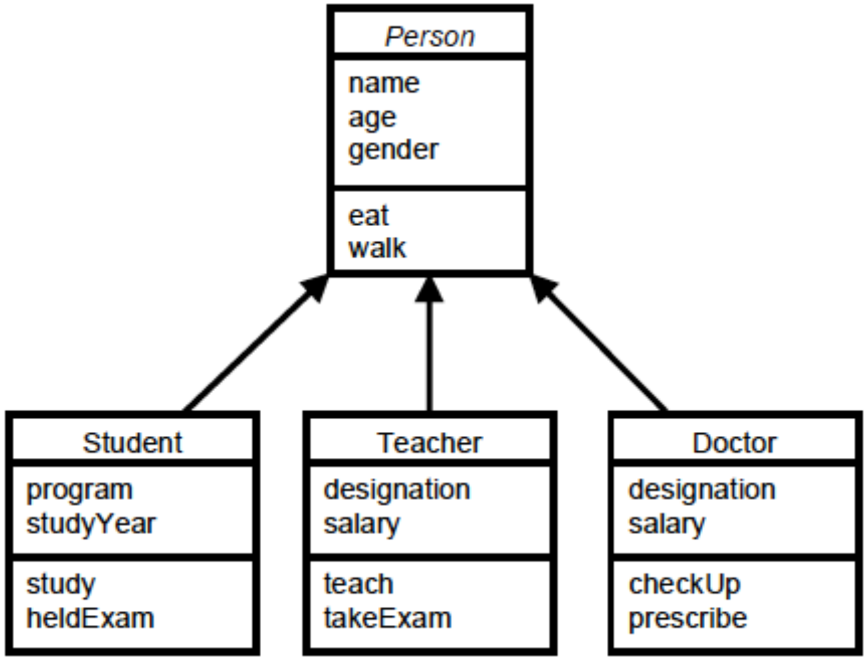
Inheritance

Basic Idea: *"Enable new objects to take on and extend properties and behavior of existing objects"*

- Inheritance is a mechanism that allows a (child or sub) class to acquire/inherit features (member variables and methods) from one or more (parent or super) classes.
- It support reusability
  - reuse the fields and methods of existing class without having to (re)write (and debug!) them.

Inheritance – “is a” Relationship

- Inheritance supports the concept of hierarchical classification, where classes are related through “is a” relationship.
  - Superclass: Person
  - Subclasses: Student, Teacher, Doctor
  - Super and all Sub classes have “ is a” relationship
    - Student is a Person
    - Teacher is a Person
    - Doctor is a Person



- Inheritance implies a **generalization/specialization** hierarchy, wherein a Subclass specializes the more general structure or behavior of its Superclass.

Polymorphism

Basic Idea: *"One interface – multiple behavior"*

- The word polymorphism comes from the Greek word for "many forms".
- Polymorphism represents a concept in type theory in which a single name, such as a variable, may denote objects of many different classes that are related by some common superclass.
- In OOP, polymorphism refers to language's ability to process variable values (e.g. in calls) differently depending on their data type or class.
- The polymorphism helps to reduce complexity by allowing the same interface/superclass to be used to specify a general class of action.
  - specifics actions (methods) are selected depending on target object

Suggested Readings

- Absolute Java, Global Edition, 6/E by Walter J. Savitch, Chap 1, Getting Started, Section “Objects and Methods”(Pages 35-36), Chap 4, Defining Classes I, Section “Class Definitions” and “Information Hiding and Encapsulation”
- Introduction to Java Programming, Brief Version, Global Edition, 11/E Liang, Chapter 9 “Objects and Classes”

- Java Tutorials
  - https://docs.oracle.com/javase/tutorial/java/concepts/index.html
  - https://docs.oracle.com/javase/tutorial/java/javaOO/index.html

Lnu.se