**OOP Basics**

* **Class** – template for object
* **Constructor** – method who create instance of class ( objects )
* **Field of class** (object) – object`s data
* **Property** – the way to get object`s data (field or attribute(
* **Methods** – DoSomething()

**What is OOP ?**

* + **Object-oriented programming** (**OOP**) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as **properties***;* and code, in the form of procedures, often known as **methods***.*
* **Encapsulation**
  + Variables of a class will be hidden from other classes
  + Accessed only through the methods of their current class

**public class Person {**

**Private Integer age;**

**Public Integer getAge() {**

**return this.age;**

**}**

**Public void setAge(int age) {**

**this.age = age;**

**}**

**}**

* **Inheritance**
  + Process where one class acquires the properties of another
  + Information is made manageable in a hierarchical order

**public class Mammal {**

**public Point move(Point x, Point y)**

**}**

**public class Person extends Mammal {**

**public void speak (String words) ( Person has property “move” )**

**}**

**public class Cat extends Mammal {**

**public void myau() ( Cat has property “move” )**

**}**

* **Polymorphism**
  + Ability of an object to take on many forms
  + Parent class reference is used to refer to a child class object

**public interface Animal {}**

**public class Mammal {}**

**public class Person extends Mammal implements Animal {}**

Person IS-A Person Person IS-A Mammal

Person IS-A Animal Person IS-A Object

**Some rules :**

* + - **Class can inherit only one class.**
    - **Class can implement many interfaces**.
* Abstraction
  + Focus on necessary context for user

**public abstract class Mammal {**

**public Point move(Point x, Point y)**

**TODO: Add moving logic**

**}**

**public class Person extends Mammal {**

**}**

**Mammal cat = new Mammal(); // can't be instantiated**

**Mammal ventsi = new Person();**

**ventsi.move();**

* + - Abstract class could be only extended.
* Cohesion :
  + Measures the strength of relationship between pieces of functionality within a given module / class.
  + It`s about how much a class depends on itself

Strong Cohesion Low Cohesion

Staff Staff

Double salary; CheckMail();

DataTime payday; sendMail();

SetSalary(newSalary) emailValidate();

getSalary() printMail();

paySalary() saveMailInDB();

* Coupling :
  + How related are two modules / classes
  + It`s about how much a class depends on the other classes.
  + Subclass Coupling
    - * The child is connected to its parent,   
        but the parent is not connected   
        to the child.
  + Temporal coupling
    - * When two actions are bundled  
         together into one module  
         just because they happen   
         to occur at the same time.