**Polymorphism**

* What is polymorphism ?
  + From the Greek

( This is something similar to word having **several** **different** meanings **depending** on the **context** )

* Polymorphism in OOP
  + Ability of an object to take on many forms

**public interface Animal {}**

**public abstract class Mammal {}**

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

**Person IS-A Person Person IS-A Animal**

**Person IS-A Mammal Person IS-A Object**

* **Reference** **type** and **Object** **type**

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

**Animal person = new Person();**

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

**Person personTwo = new Person();**

Reference type Object Type

* Variables are saved in reference type
* You can use only reference methods
* If you need object method you need to cast it or override it
* Types of Polymorphism
  + Runtime polymorphism
    - Method overriding

public class Shape {}

public class Circle extends Shape {}

public static void main(String[] args) {

**Shape** shape = new **Circle**()

}

* + Compile time polymorphism
    - Method overloading

public static void main(String[] args) {

**int sum**(int a, int b, int c)

**double** **sum**(Double a, Double b)

}

* + - Also known as **Static Polymorphism**

public static void main(String[] args) {

**static int myMethod**(int a, int b) {}

**static Double myMethod**(Double a, Double b)

}

* + - Argument lists could differ in:
      * Number of parameters
      * Data type of parameters
      * Sequence of Data type of parameters
* Rules for **overloading method**
  + **Overloading can take place in the same class or in its sub-class**
  + **Constructor in Java can be overloaded**
  + **Overloaded methods must have a different argument list**
  + **Overloaded methods should always be the part of the same class ( can also take place in sub class), with same name but different parameters**
  + **They may have the same or different return types**
* Rules for **overriding method**
  + **Overriding can take place sub-class**
  + **Argument list must be the same as that of the parent method**
  + **The overriding method must have same return type**
  + **Access modifier cannot be more restrictive**
  + **Private, static and final methods can NOT be overridden**
  + The overriding method **must not** throw new or broader **checked exceptions**
* **Abstract Classes**
  + **Abstract class can not be instantiated**

public abstract class Shape {}

public class Circle extends Shape {}

Shape shape = new Shape(); // Compile time error

Shape circle = new Circle(); // polymorphism

* + An abstract class may or may not include abstract methods.
  + If it has at least one abstract method, it must be declared abstract
  + To use abstract class, you need to extend it