Agenda

- this reference
- Method Overloading
- Types of Methods
- Constructor Chaning
- ~~Array~~

this Reference

- "this" is implicit reference variable that is available in every non-static method of class which is used to store reference of current/calling instance
- Whenever any non-static method is called on any object, that object is internally passed to the method and internally collected in implicit "this"
- "this" is constant within method i.e. it cannot be assigned to another object or null within the method.
- Using "this" inside method (to access members) is optional.
- However, it is good practice for readability.
- In a few cases using "this" is necessary.

Types of Methods

1. constructor
2. setters
    - Used to set value of the field from outside the class.
    - It modifies state of the object.
3. getters
    - Used to get value of the field outside the class.
4. facilitators
    - Provides additional functionalities
    - Business logic methods

Constructor

- It is a special method of the class
- In Java fields have default values if unitialized
- Primitive types default value is usually zero
- Reference type default value is null
- Constructor should initialize fields to the desired values.
- Types of Constructor
    - 1. Default/Parameterless Ctor
    - 2. Parameterized Ctor

Constructor Chaning

- Constructor chaining is executing a constructor of the class from another constructor (of the same class).
- Constructor chaining (if done) must be on the very first line of the constructor.

Object/Field Initializer

- In C++/Java Fields of the class are initialized using constructor
- In java, field can also be initialized using
    - 1. field initializer
    - 2. object initializer
    - 3. Constructor

Method Overloading

- Defining methods with same name but differnt arguments(signature) is called as method overloading
- Arguments can differ in one of the following ways

1. No of parameters should be different
2. If no of parameters are same then their type of parameters should be different
3. If no and type are same then the order of paramters should be different

- Count (no of parameters)

```
static int multiply(int a, int b) {
return a * b;
}
static int multiply(int a, int b, int c) {
return a * b * c;
}
```

- type of parameter

```
static int square(int x) {
return x * x;
}
static double square(double x) {
return x * x;
}
```

- Order or parameters

```
static double divide(int a, double b) {
return a / b;
}
static double divide(double a, int b) {
return a / b;
}
```

- Note that return type is NOT considered in method overloading.