Agenda

- this reference
- Types of Methods
- Constructor Chaning
- Array
- ~~Variable Arity/Argument Method~~
- ~~Object/Field Initializer~~

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 like.

1. Unhide non-static fields from local fields i.e to differentiate between local variable and field of the class.

```
double height; // "height" field
 void setHeight(double height) { // "height" parameter
    this.height = height;
}
```

2. Constructor chaining
3. Access instance of outer object.

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.

Array

- Array is collection of similar data elements. Each element is accessible using indexes
- It is a reference type in java
- its object is created using new operator (on heap).
- The array of primitive type holds values (0 if uninitialized) and array of non-primitive type holds references (null if uninitialized).
- In Java, checking array bounds is responsibility of JVM. When invalid index is accessed, ArrayIndexOutOfBoundsException is thrown.
- Array types are
    - 1. 1-D array
    - 2. 2-D/Multi-dimensional array
    - 3. Ragged array
        - In 2D array if the second dimension of array is having differnt length then such array is called as Ragged Array
- Array of primitive types

```
int[] arr1 = new int[5];
int[] arr2 = new int[5] { 11, 22, 33 }; // error
int[] arr3 = new int[] { 11, 22, 33, 44, 55 };
int[] arr4 = { 11, 22, 33, 44 };
```

- Array of non-primitive types

```
Human[] arr5 = new Human[5];
Human[] arr6 = new Human[] {h1, h2, h3};  // h1, h2, h3 are Human references
Human[] arr7 = new Human[] {
new Human(...),
new Human(...),
```

```
new Human(...)
};
```

1. 1-D array

- Individual element is accesses as arr[i].

```
int[] arr = new int[5];
int[] arr = new int[4] { 10, 20, 30, 40 };  // error
int[] arr = new int[] { 11, 22, 33, 44, 55 };
int[] arr = { 11, 22, 33, 44 };
Human[] arr = new Human[5];
Human[] arr = new Human[] {h1, h2, h3};
```

2. 2-D/Multi-dimensional array

- 2-D/Multi-dimensional array
- Internally 2-D arrays are array of 1-D arrays. "arr" is array of 2 elements, in which each element is 1-D array of 3 doubles.
- Individual element is accesses as arr[i][j].

```
double[][] arr = new double[2][3];
double[][] arr = new double[][]{ { 1.1, 2.2, 3.3 }, { 4.4, 5.5, 6.6 } };
double[][] arr = { { 1.1, 2.2, 3.3 }, { 4.4, 5.5, 6.6 } };
```

3. Ragged Array

- Ragged array is array of 1-D arrays. Each 1-D array in the ragged array may have different length.

```
int[][] arr = new int[4][];
arr[0] = new int[] { 11 };
arr[1] = new int[] { 22, 33 };
arr[2] = new int[] { 44, 55, 66 };
arr[3] = new int[] { 77, 88, 99, 110 };

for(int i=0; i<arr.length; i++) {
for(int j=0; j<arr[i].length; j++) {
        System.out.print(arr[i][j] + ", ");
    }
}
```