Object Oriented Programming with Java II

Dr. Mohamed K. Hussein

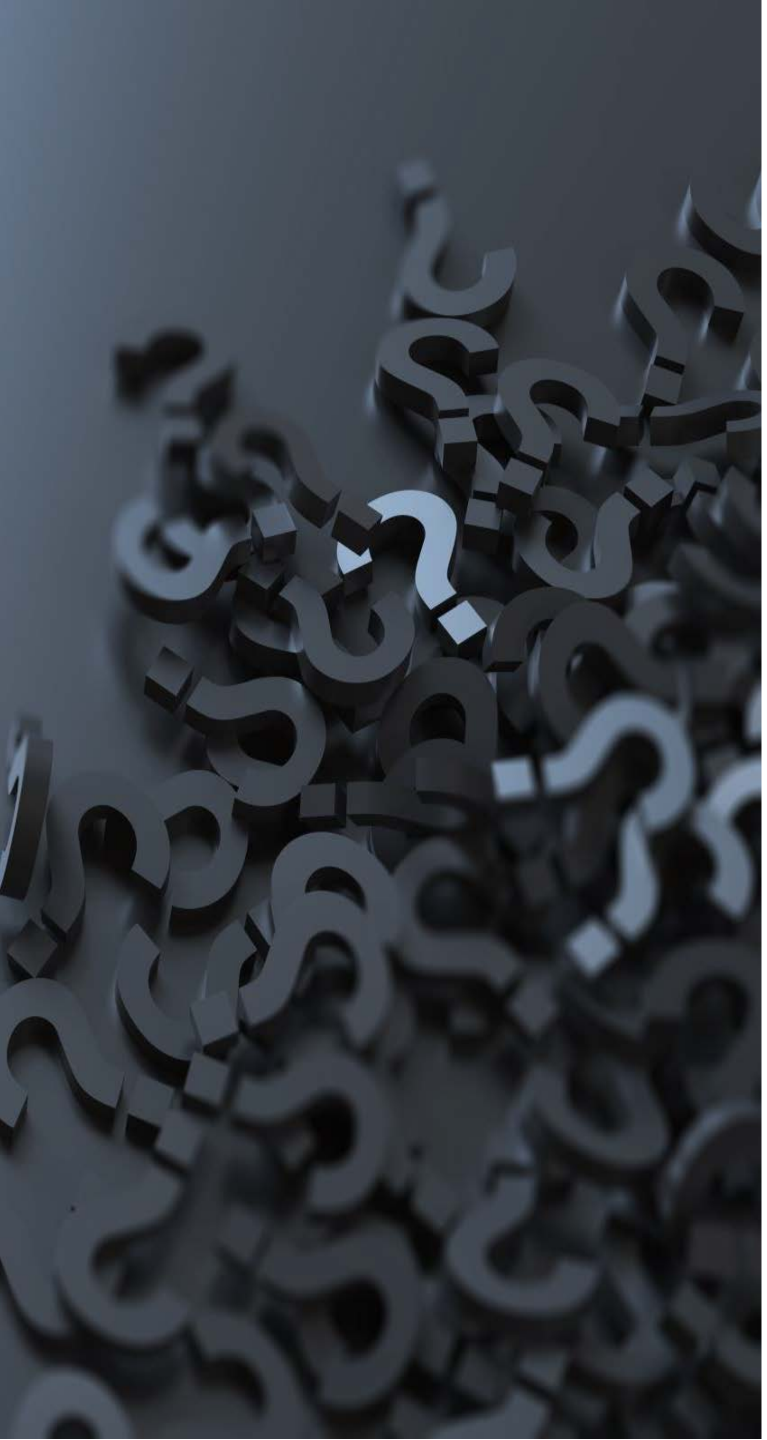
Lecture 3

Associate Prof., Computer Science department,

Faculty of Computers & Information Science,

Suez Canal University

Email: m_khamis@ci.suez.edu.eg



Lecture outcomes

- The *this* reference
- Array of objects
  - Passing array of objects to methods
- Static variables and methods
  - Usage and syntax
- Passing object to the constructor
- Immutable objects and classes

The *this* Reference

- The *this* reference allows an object to refer to itself.
  - That is, the this reference, used inside a method, refers to the object through which the method is being executed

- The *this* reference can also be used to distinguish the parameters of a constructor from the corresponding instance variables with the same names

```
public Person (int age, Sring name)
{
   this.age = age;
   this.name = name;
}
```

Array of Objects

- Although referred to as an array of objects they are actually arrays of references to objects.
- Recall for arrays: 2 steps are involved to create the array

```
int [] array;              // Reference to array
array = new int[3];        // Creates array of integers
```

- Recall for objects: 2 steps are required to create the object

```
Person ahmed;              // Reference to Person object
ahmed = new Person();      // Creates object
```

Array of Objects

- An array of objects is actually an array of references to objects.
- So 3 steps are usually required
  - Two steps are still needed to create the array

```
// Step 1: create reference to array
Person [] somePeople;

// Step 2: create array
somePeople = new Person[3];
```

  - In Java after these two steps each array element will be null.

```
 somePeople[0].setAge(10);  // Null pointer exception
```

Array of Objects

- The third step requires traversal through array elements (as needed):
  - create a new object and have the array element refer to that object.

```
for (i = 0; i < 3; i++)
{
   // Create object, array element refers to that object
   somePeople[i] = new Person();

   // Now that array element refers to an object, a method can be called.
   somePeople[i].setAge(i);
}
```

Passing/return array of objects to/from methods

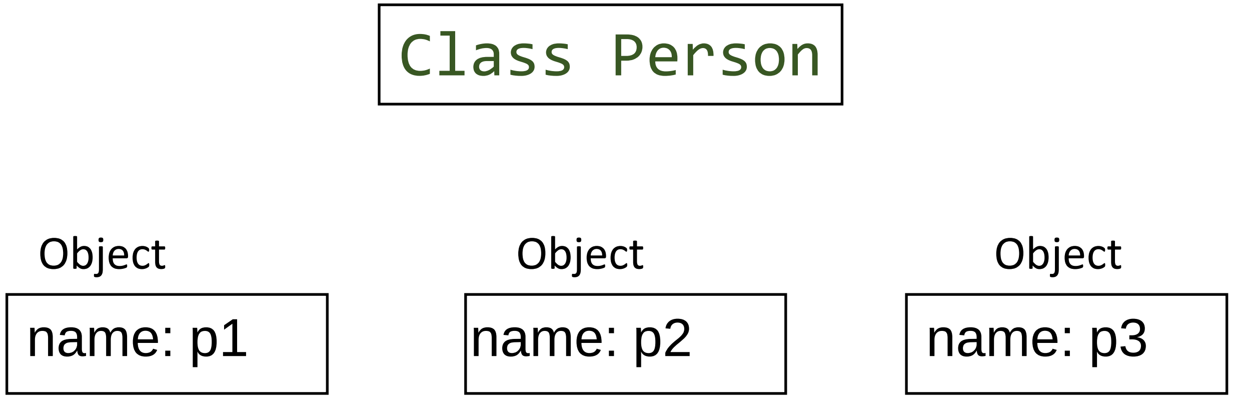
```
public class MainClass{
     public static void main(String [] args) {
        Person [] somePeople;                      // Reference to array
        somePeople = createObjectArray();          // Create array
        printArray(somePeople);
     }
    public static Person[] createObjectArray() {
        Person[] people =  new Person[5];
        for (int i = 0; i < people.length; i++) {
         people[i] = new Person();
         People[i].setAge(i*10);
        }
        return people;                             // Return array of objects
    }
    public static void printArray( Person[] people) {
       for (int i = 0; i < people.length; i++)
        System.out.println("Age " +  people[i].getAge());
    }
}
```

Static Variables

- Static variables are also called *class variables*
- Normally, each object has its own data space, but if a variable is declared as static, only one copy of the variable exists

```
private static int number;
```

- Memory space for a static variable is created when the class in which it is declared is loaded
- All objects created from the class share static variables
- Changing the value of a static variable in one object changes it for all others



Static Variables

- Inside the class definition

Format:

```
<Access permission> static <attribute or method name>
```

Example:

```
class Person
{
    private static int number = 0;

    public Person()
    {
        number++;
    }
}
```

Static Methods

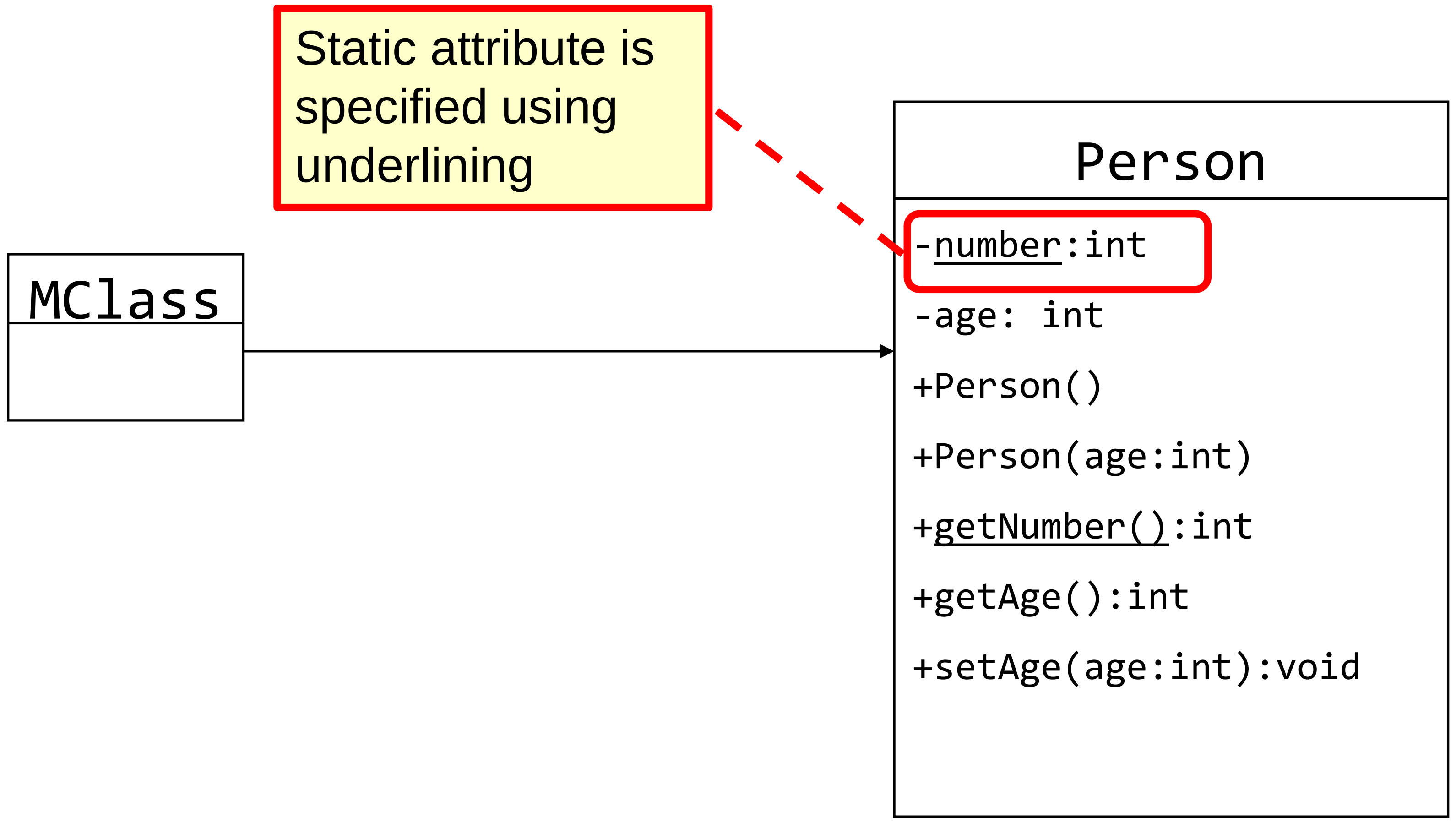
- Associated with the class as a whole and not individual instances of the class.
  - Can be called without having an instances (because it's called through the class name not a reference/instance name).

```
double squareRoot = Math.sqrt(9);        // ClassName.method()
```

  - ➢ Recall that the main method is static;
    - ➢ it is invoked by the system without creating an object

- Static methods cannot reference instance variables, because instance variables don't exist until an object exists
  - ➢ However, a static method can reference static variables or local variables

Static Data & Methods: UML Diagram



Passing object to the constructor

```
public class Person{
    private int age;
    public Person(){age = 0;}
    public Person(int age){
        this.age = age;
    }
    public Person(Person p){
        this.age = p.getAge();
    }
    public int getAge(){
        return age;
    }
    public void setAge(int age){
        this.age = age;
    }
}
```

```
public class Main{
    public static void main(String[] args) {
        Person p1 = new Person(20);
        Person p2 = new Person(p1);
        p1.setAge(30);
        System.out.println(p1.getAge());
        System.out.println(p2.getAge());
    }
}
```

Immutable objects & classes

- Define immutable classes to create immutable objects.
  - The contents of immutable objects cannot be changed.

- <u>Required steps to create immutable class:</u>
  - All the attributes must be private
  - Cannot contain public setter methods for any data fields.
  - ***No accessor methods can return a reference to a data field that is mutable.***

```
import java.util.Date;public
class Person{
    private int age;
    private Date dateCreated;
    public Person(){
        age = 0;
        dateCreated = new Date();
    }
    public Date getDate(){
         return dateCreated;
    }
}
```

```
import java.util.Date;
public class Main{
    public static void main(String[] args) {
        Person p1 = new Person();
        Date d1;
        System.out.println(p1.getDate());
        d1 = p1.getDate();
        d1.setDate(200000);
        System.out.println(p1.getDate());
    }
}
```

Assignment 3

I. Design a Shopping Cart program. In this task you will complete a class that implements a shopping cart as an array of items. The Item class models an item one would purchase. An item has a name, price, and quantity (the quantity purchased). The file ShoppingCart.java implements the shopping cart as an array of Item objects. Complete the ShoppingCart class by doing the following:

- Declare an instance variable cart to be an array of Items and instantiate cart in the constructor to be an array holding capacity Items.
- There should be addToCart method. This method should add the item to the cart and update the totalPrice instance variable (note this variable takes into account the quantity).

Assignment 3

II. Design a library program. In this task you will complete a class that implements a Library as an array of Books. The Book class that models a book one would find the library. A book has a title, price, year, and Author. The Library.java implements the library as an array of Books objects. Complete the Library class by doing the following:

- Declare an instance variable library to be an array of Books and instantiate library in the constructor to be an array holding capacity Items.
- There should be addBook method. This method should add the book to the library.

Thank you