Android Development

Ch-00

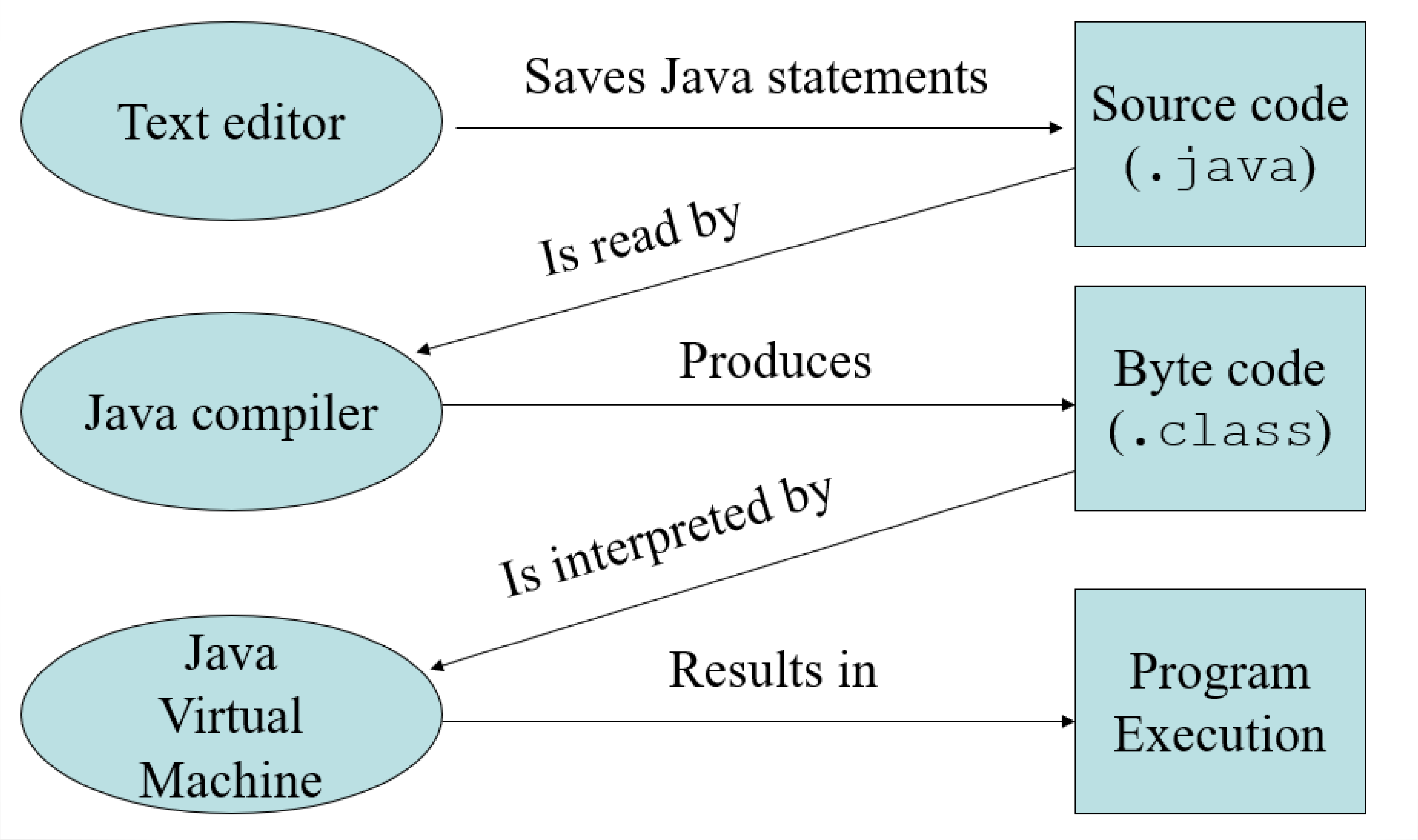
Programming Concepts

- A **program** is a set of instructions a computer follows in order to perform a task. These statements of instructions are known as source code.
- An **algorithm** is a set of well defined steps to completing a task.
  Ex: sum 2 numbers and print the result.
- **Syntax errors** are mistakes violate the rules of the programming language.
- A **compiler** is a program that translates source code into an executable form. Will show the results during compiling. And stop the results where there’s error. Like in dev-c++.
- **Interpreter:** reads all code lines and break where founds error. After finishing reading will show the result.

Programming Concepts

- **Byte code** files end with the .class file extension.
- Byte code instructions are the machine language of the Java Virtual Machine (JVM), cannot be directly executed by the CPU.
- **JVM** is often called an interpreter. using it the deferent devices can understands the java source code.
- **Object**: every thing can be considered in the world and has attributes. It’s a combination of data and procedures that manipulate that data. This name because we can’t call it string, int, … .
- **Data**: raw facts about objects.
- **Information**: the result of processing the data of objects. Ex: (Name, Grade), (Name, Length).

Program Development Process

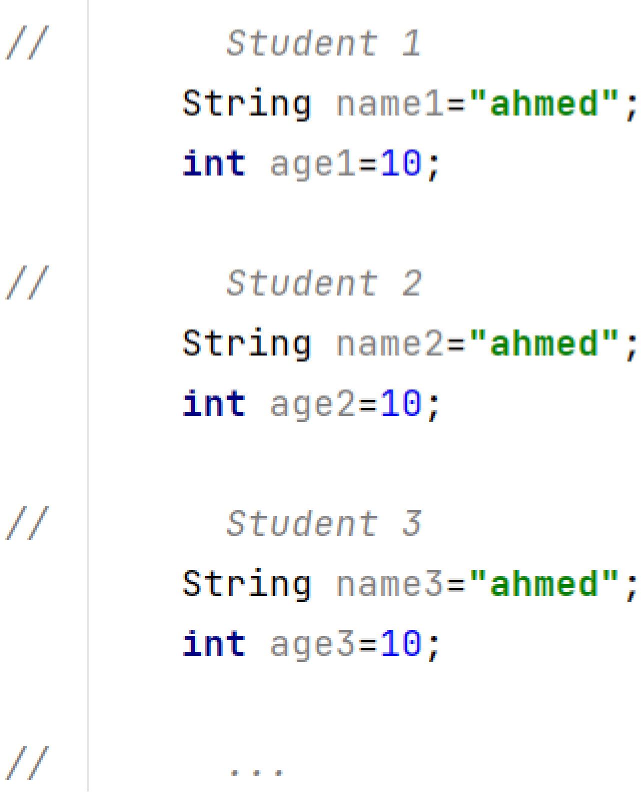


Introduction to OOP

- Suppose there is a school with 100 students that wants to save its students' personal data, a copy of their marks of academic exams, how does the school keep this data in an organized manner?
- Suppose we want to simulate this system in application, how does it saves this data in an organized manner?

Introduction to (OOP)

- **Bad Way:**

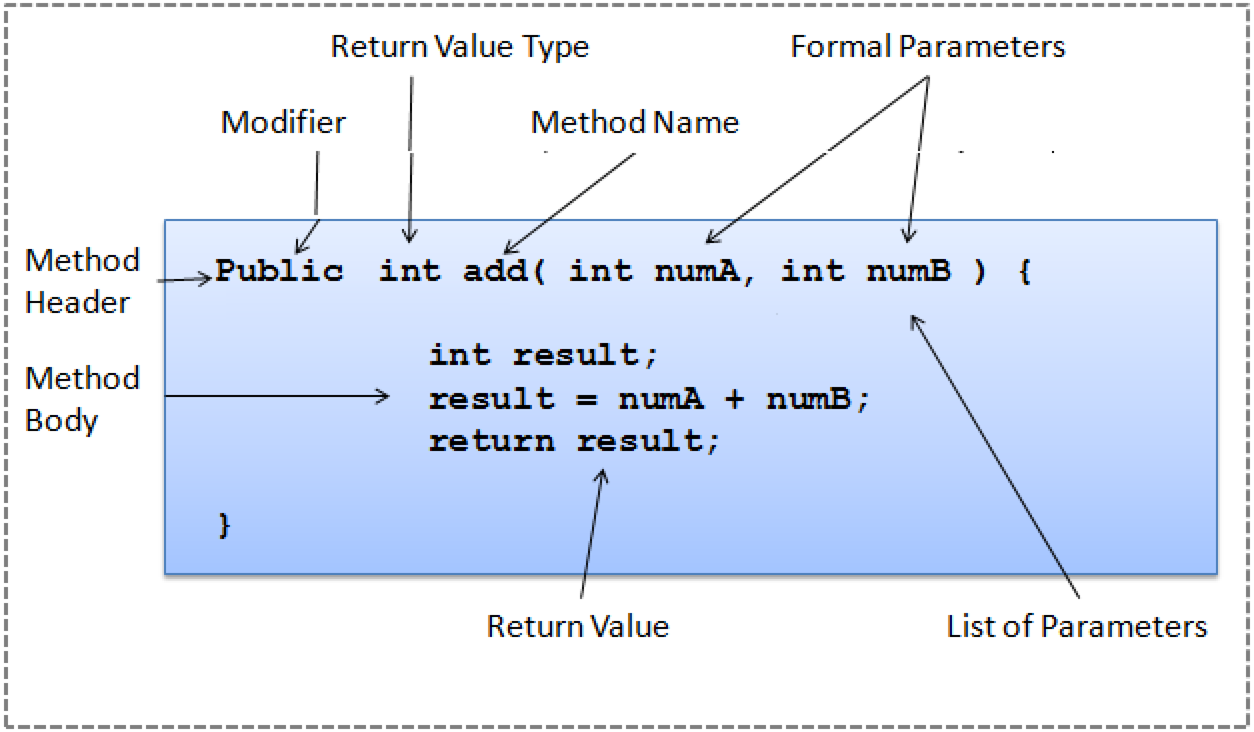


```
//        Student 1
        String name1="ahmed";
        int age1=10;

//        Student 2
        String name2="ahmed";
        int age2=10;

//        Student 3
        String name3="ahmed";
        int age3=10;

//        ...
```

Programming Concepts (OOP)

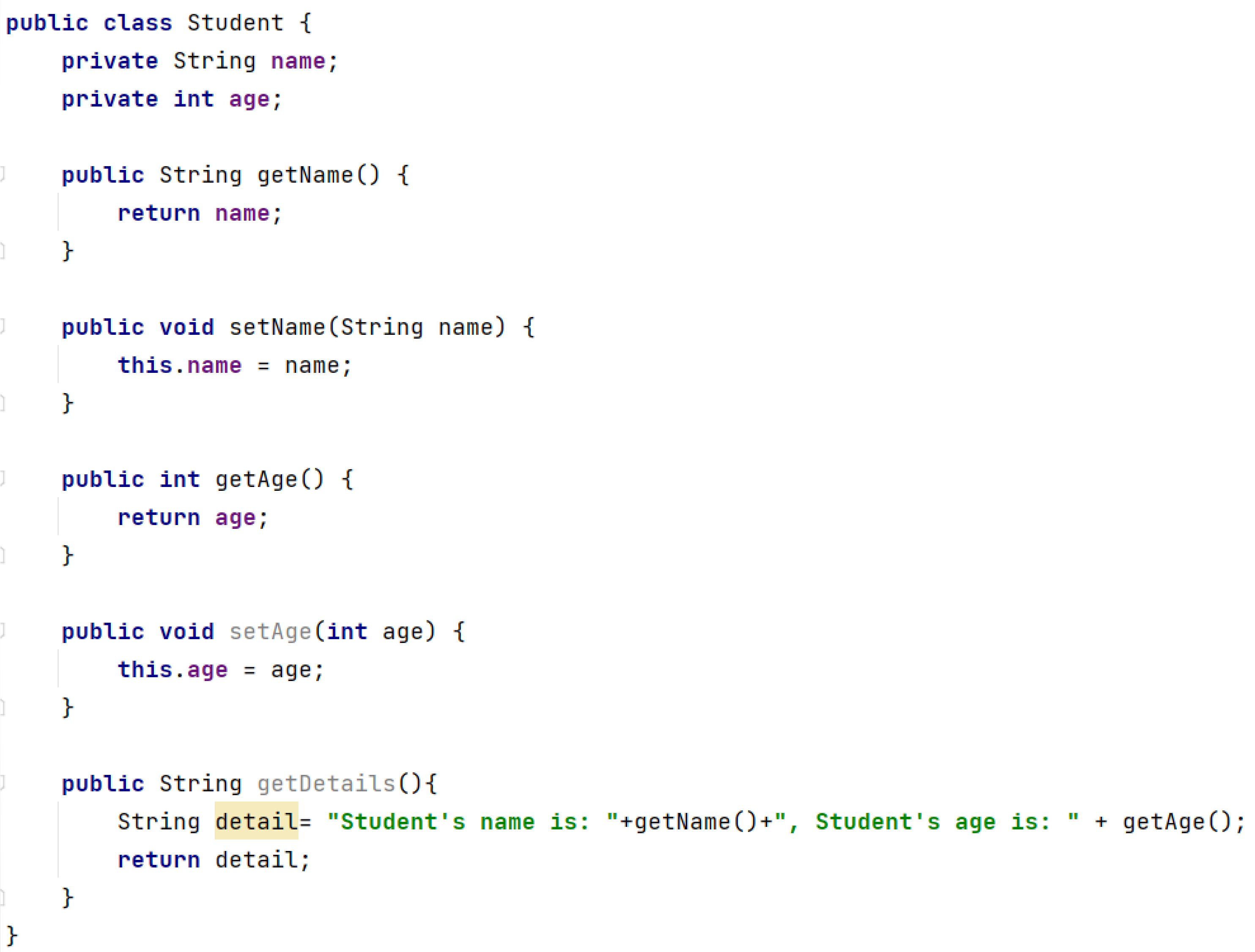
- **Object-Oriented Programming(OOP)**
- **OOP** is a programming paradigm that allows you to package together object’s data and procedures to modify those data, while keeping the details hidden away.
- Data in an object are known as **attributes**.
- Procedures in an object are known as **methods**.
- Object-oriented programming combines data and methods via ***Encapsulation****, this encapsulation called as:* **Class**.
- **Class:** is a container that has many variables and methods related to each other, used to describe objects.
- Only an objects methods should be able to directly manipulate its attributes.

Programming Concepts (Encapsulation)



```
Public  int add( int numA, int numB ) {

          int result;
          result = numA + numB;
          return result;

}
```

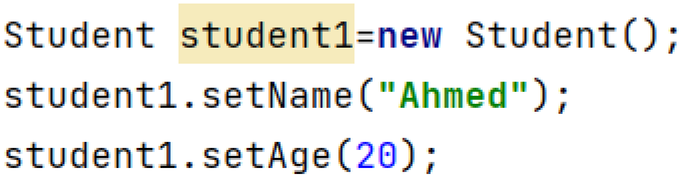
Class ex.



```java
public class Student {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getDetails(){
        String detail= "Student's name is: "+getName()+", Student's age is: " + getAge();
        return detail;
    }
}
```

Create object ex.

- How we create an instance of object class?



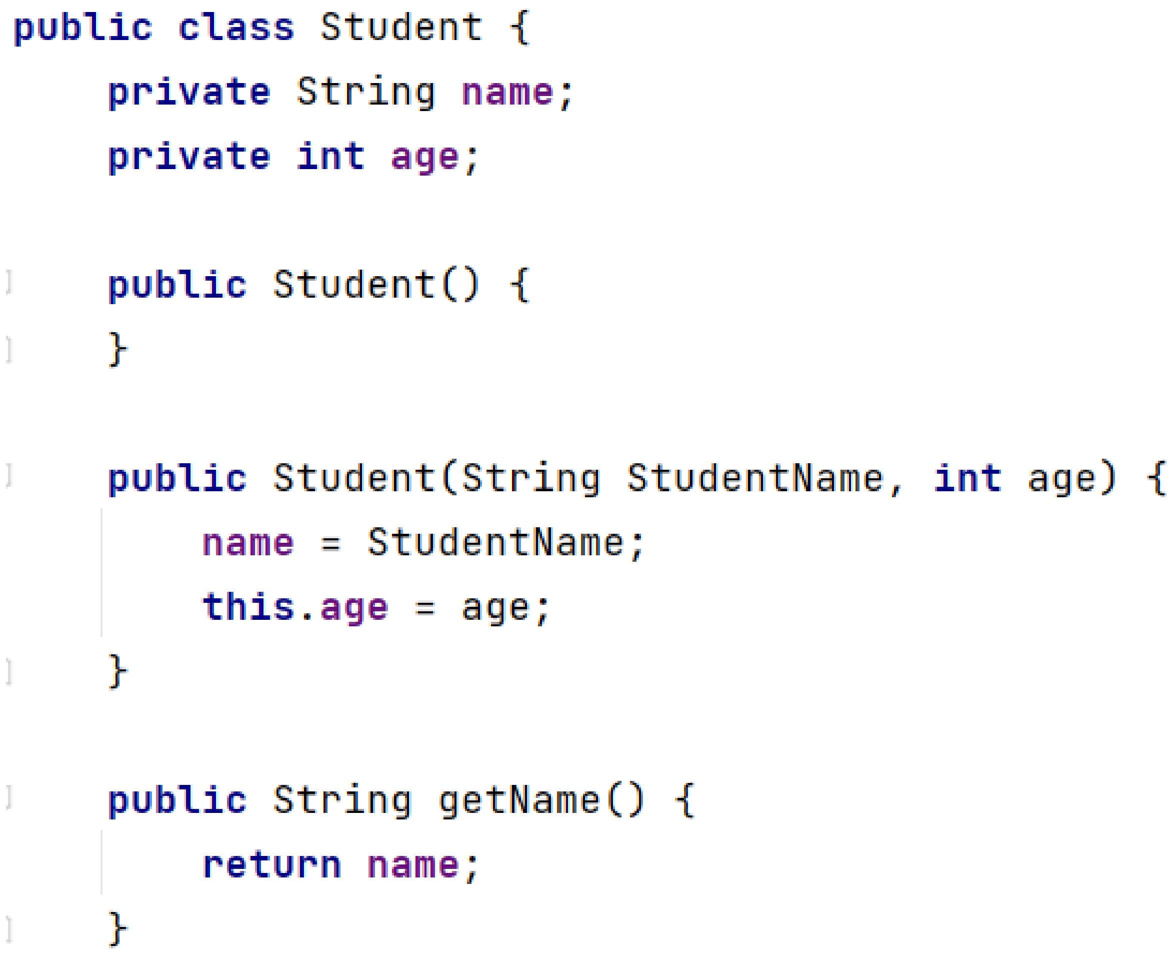
```
Student student1=new Student();
student1.setName("Ahmed");
student1.setAge(20);
```

- Here we create student1 variable, its type: Student class.
- New Student();
- Here we allocate a space in the memory to save student1 data.
- student1: points to its location in the memory.
- By writing *student.* We access the methods into the class.
- **Note:** we don’t save the data into a class, but in the object of a class that has a space in memory.

Programming Concepts (OOP)

- **Constructor** in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. we can use it to give the object default values for it. Each class must contain at least one constructor.
- Constructors are almost similar to methods except for two things
  - its name is the same as the class name.
  - it has no return type.
- The class can contain 1 or more constructors with deferent arguments.

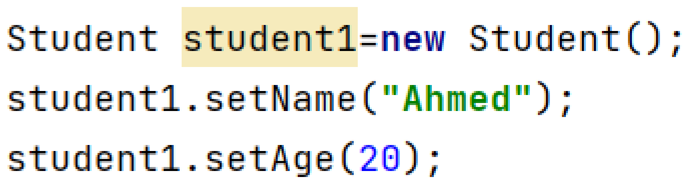
Programming Concepts (class)



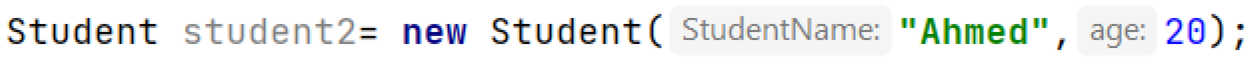
```java
public class Student {
    private String name;
    private int age;

    public Student() {
    }

    public Student(String StudentName, int age) {
        name = StudentName;
        this.age = age;
    }

    public String getName() {
        return name;
    }
```

Create object ex.

- How we create an instance of student object class?



```
Student student1=new Student();
student1.setName("Ahmed");
student1.setAge(20);
```



```
Student student2= new Student( StudentName: "Ahmed", age: 20);
```

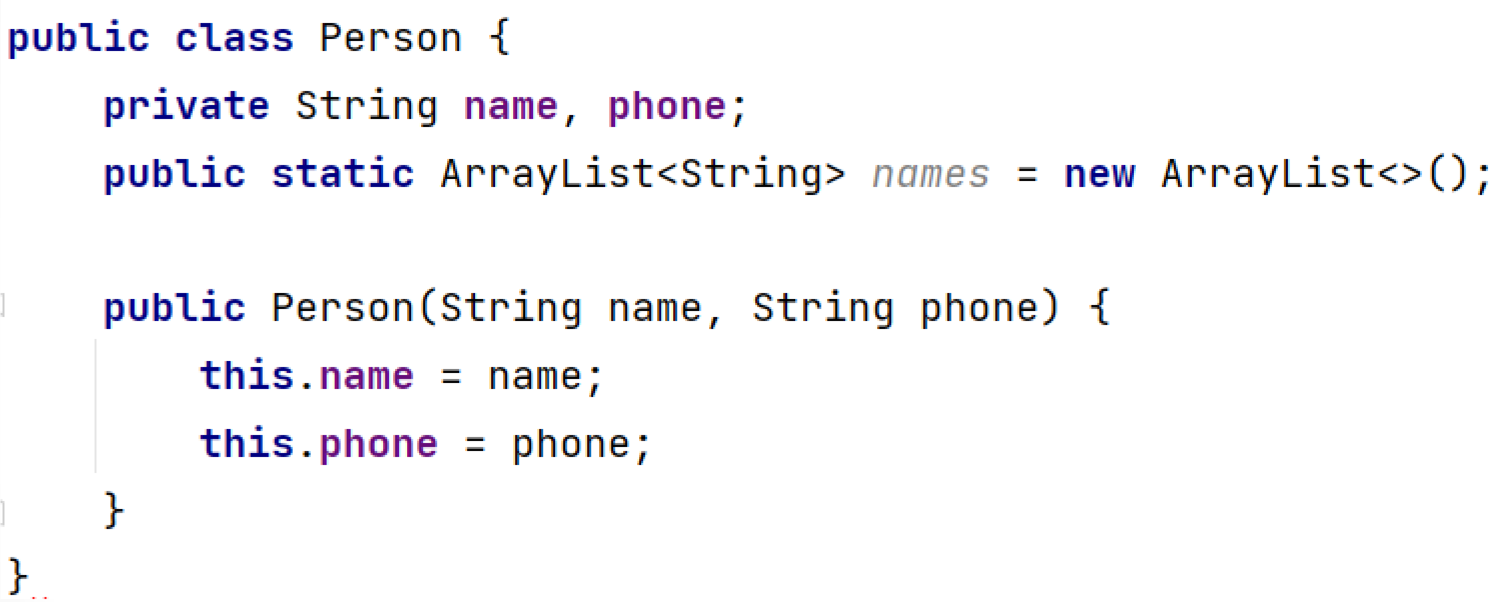
- At student1: we depends on default empty constructor.

- At student2: we depends on the second constructor.

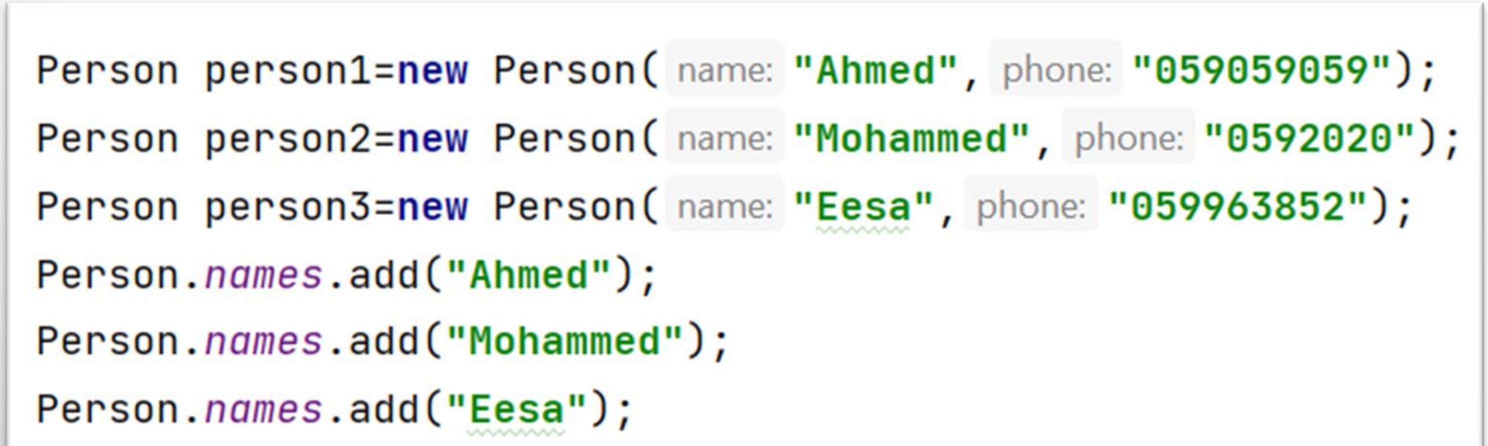
Programming Concepts (static)

- The static keyword in Java is used for memory management mainly (it saves memory).
- We can apply static keyword with :
  variables (known as a class variable).
  methods (known as a class method).
  nested classes.
- We deal with it when the member is expected to be used a lot.
- Static members are saved in the memory first of all Regardless of creating instances / objects of it.
- We call them by class Name.

Example of static



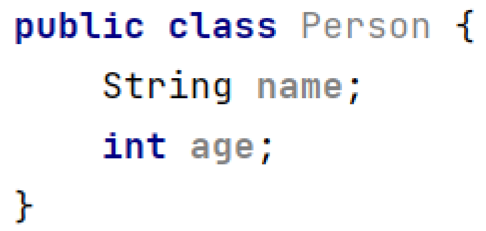
```
public class Person {
    private String name, phone;
    public static ArrayList<String> names = new ArrayList<>();

    public Person(String name, String phone) {
        this.name = name;
        this.phone = phone;
    }
}
```



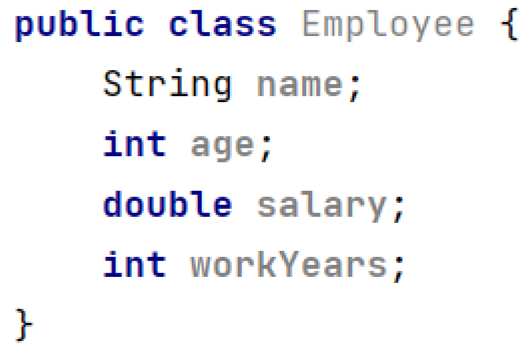
```
Person person1=new Person( name: "Ahmed", phone: "059059059");
Person person2=new Person( name: "Mohammed", phone: "0592020");
Person person3=new Person( name: "Eesa", phone: "059963852");
Person.names.add("Ahmed");
Person.names.add("Mohammed");
Person.names.add("Eesa");
```

Programming Concepts (Inheritance)

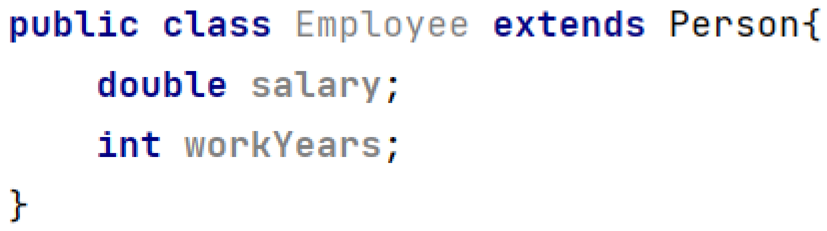
- It is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods, there are superclass and child class. **Ex:**



```java
public class Person {
    String name;
    int age;
}
```



```java
public class Employee {
    String name;
    int age;
    double salary;
    int workYears;
}
```

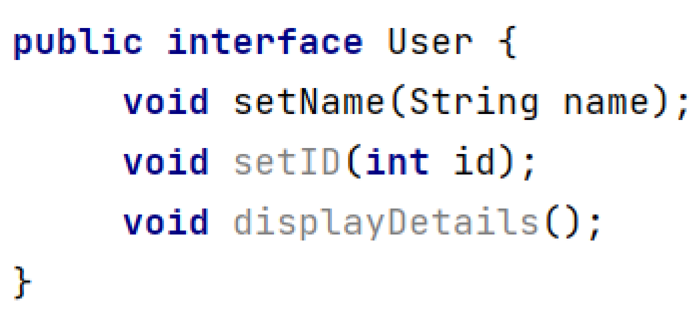


```java
public class Employee extends Person{
    double salary;
    int workYears;
}
```

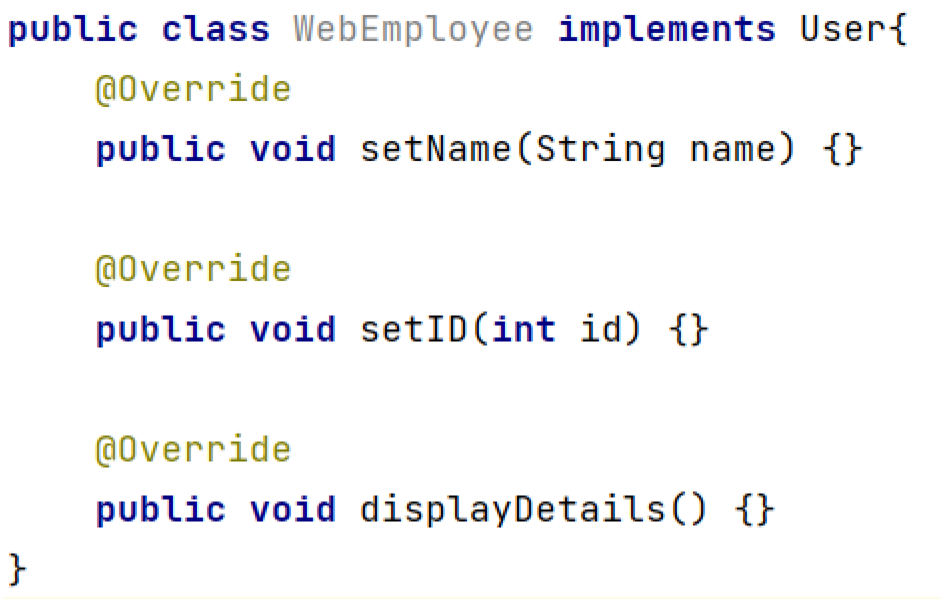
Programming Concepts (interface class)

- In an interface, you can’t instantiate variables and create an object. But constant variables.
- All of its methods are without body.
- The classes that inherits this class this class must override its functions.

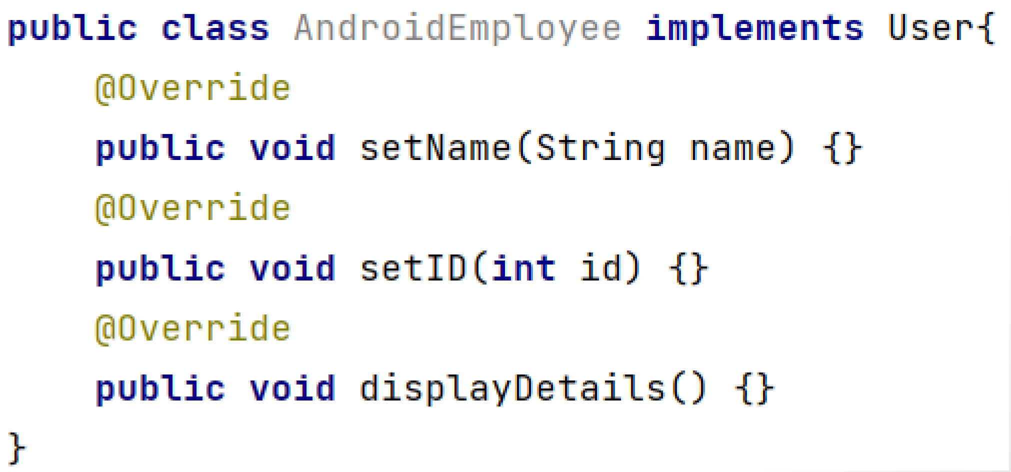
Interface class ex.



```
public interface User {
    void setName(String name);
    void setID(int id);
    void displayDetails();
}
```



```
public class WebEmployee implements User{
    @Override
    public void setName(String name) {}

    @Override
    public void setID(int id) {}

    @Override
    public void displayDetails() {}
}
```



```
public class AndroidEmployee implements User{
    @Override
    public void setName(String name) {}
    @Override
    public void setID(int id) {}
    @Override
    public void displayDetails() {}
}
```

Done

Fares Saleem