Object Oriented Programming

ICT2122

Classes and Objects

P.H.P. Nuwan Laksiri
Department of ICT
Faculty of Technology
University of Ruhuna

Lesson 02

Recap

- What is Object Oriented Programming
- Fundamentals of Object Orientation
- Why Object Orientation ?
  - Modularity
  - Information-hiding
  - Code re-use
  - Pluggability and debugging ease
- Understanding Classes and Objects
- Real-World Scenario
- Class
- Object
- Instance
- Instantiation

Outline

- Object Oriented Programming - Concepts
- Understanding Objects
- Understanding Classes
- Understanding Fields
- Understanding Methods
- JAVA - Access Modifiers
- Creating Objects
- Initializing Objects
  - By reference variable
  - By method
  - By constructor
- Understanding Constructors
  - Default
  - Parameterized

Object Oriented Programming - Concepts

- Object Oriented Programming simplifies the software development and maintenance by providing some concepts
- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Classes and Objects

A class is like a cookie cutter; it defines the shape of objects

Objects are like cookies; they are instances of the class

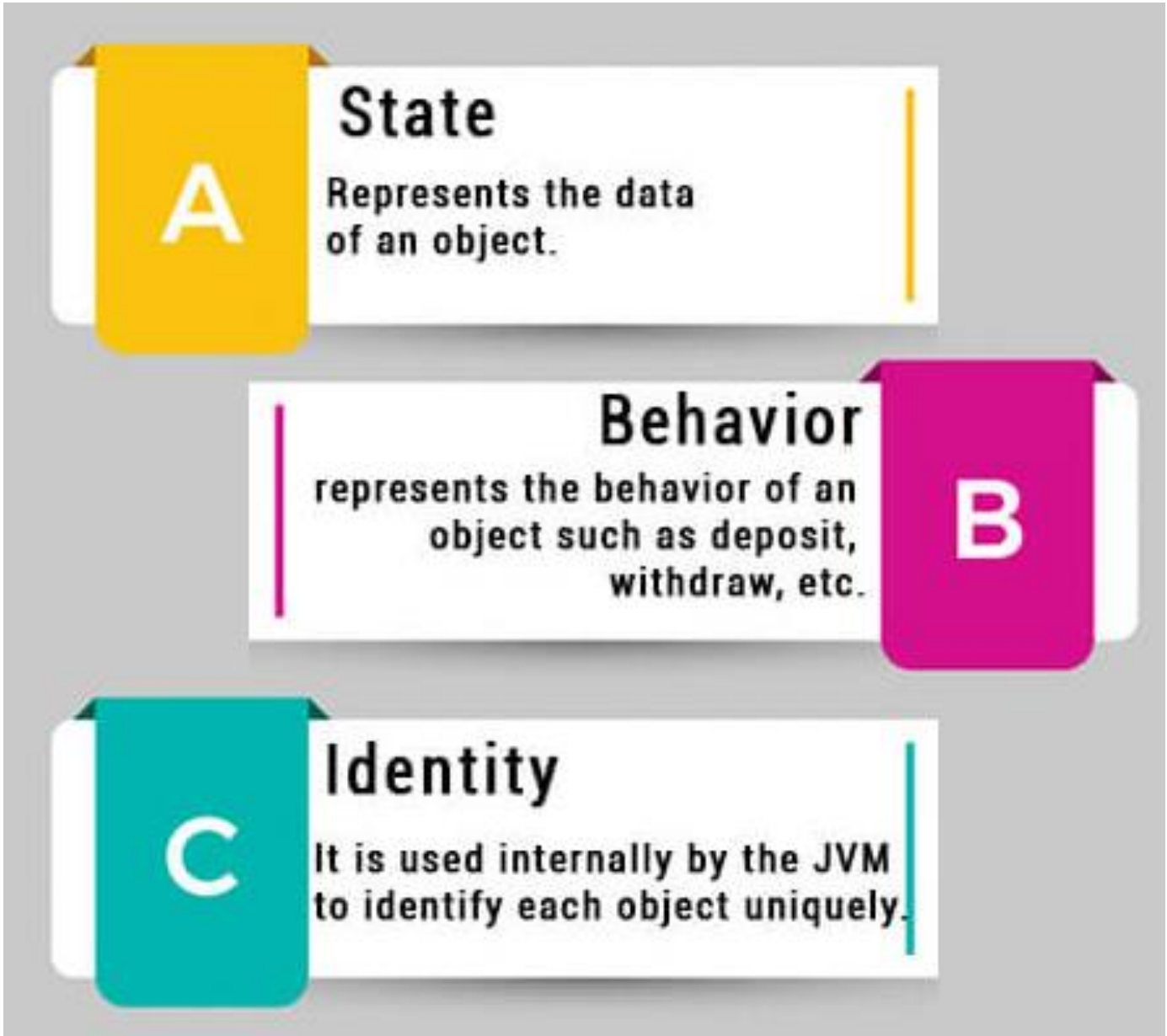
Photograph courtesy of Guillaume Brialon on Flickr.

What Is an Object?

- Object is an instance of a class.

https://docs.oracle.com/javase/tutorial/java/concepts/object.html

Characteristics of an Object



Understanding Objects

- Objects have State
- Objects have Behavior
- Objects have Identity
- Objects have Type

Understanding the Life Cycle of an Object

1. Before an object can be created from a class, the class must be loaded.
2. An object is created from a class when you use the new keyword.
3. The object lives its life, providing access to its public methods and fields to whoever wants and needs them.
4. When it's time for the object to die, the object is removed from memory, and Java drops its internal reference to it.

What Is a Class

- A blueprint of an object.

https://docs.oracle.com/javase/tutorial/java/concepts/class.html

Declaring a Class

- All classes must be defined by a class declaration — lines of code that provide the name for the class and the body of the class.

Basic Form

```
class  ClassName
{
        class body
}
```

Declaring a Class

Extended Form

Package declaration;

Import statements;

[access modifier] class ClassName extends [ClassName] implements [Interface Names]

{

Fields

Methods

Constructors

Initializers

other Classes and Interfaces

}

Body of a Class

- Fields:
  - Variable declarations define the fields of a class.
- Methods
  - Method declarations define the methods of a class.
- Constructors
  - A constructor is a block of code that's similar to a method but is run to initialize an object when an instance is created.
- Initializers
  - These stand-alone blocks of code are run only once, when the class is initialized.
  - static initializers and instance initializers
- Other classes and interfaces
  - A class can include another class, which is then called an inner class or a nested class. Classes can also contain interfaces.

Homework

- Who are the *members* of a class?

Ordering Elements in a Class

Element	Example	Required?	Where does it go?
Package declaration	`package abc;`	No	First line in the file
Import statements	`import java.util.*;`	No	Immediately after the package
Class declaration	`public class C`	Yes	Immediately after the import
Field declarations	`int value;`	No	Anywhere inside a class
Method declarations	`void method()`	No	Anywhere inside a class

Class Naming Convention

- Begin the class name with a capital letter
  - Ex : Student, TennisBall
- Use nouns for your class names as much as possible
- Try to avoid using the names of a Java Keywords, API class names, Reserved words etc.

Class – How to Save

- A public class must be written in a source file that has the same name as the class, with the extension java.
  - A public class named Student → Student.java

- You can't place two or more public classes in the same file.
  - Let's tryout
    - Including two public classes in the same file
    - Including one public class and another nonpublic class

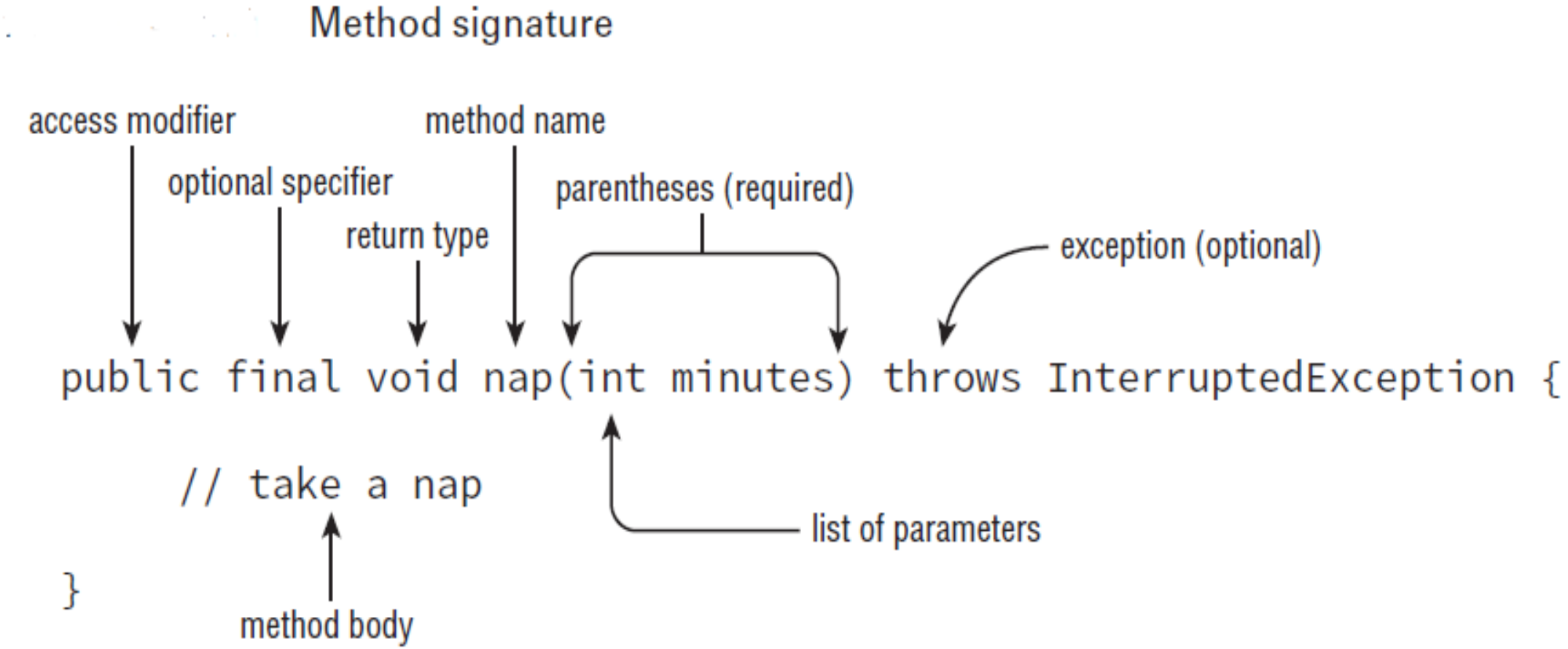
Understanding Fields

- A field is a variable that's defined in the body of a class, outside any of the class's methods.
- Fields, which are also called class variables, are available to all the methods of a class.

Understanding Methods

- A Java method is a collection of statements that are grouped together to perform an operation.
- A method only runs when it is called.

Understanding Methods



Parts of Method declaration

Element	Value in nap() example	Required?
Access modifier	`public`	No
Optional specifier	`final`	No
Return type	`void`	Yes
Method name	`nap`	Yes
Parameter list	`(int minutes)`	Yes, but can be empty parentheses
Optional exception list	`throws InterruptedException`	No
Method body	`{ // take a nap }`	Yes, but can be empty braces

JAVA - Access Modifiers

Java offers four choices of access modifiers:

- **public**
  - The method can be called from any class.
- **private**
  - The method can only be called from within the same class.
- **protected**
  - The method can only be called from classes in the same package or subclasses.
- **Default**
  - (Package Private) Access The method can only be called from classes in the same package.
  - This one is tricky because there is no keyword for default access. You simply omit the access modifier.

Creating Objects

- **Declaration**:
  - Variable declarations that associate a variable name with an object type.
- **Instantiation**:
  - The new keyword is a Java operator that creates the object.
- **Initialization**:
  - The new operator is followed by a call to a constructor, which initializes the new object.

https://docs.oracle.com/javase/tutorial/java/javaOO/objectcreation.html

Declaring and Instantiating(Creating) an Object

- The "**new**" keyword is used to instantiate an object.
- This will create the object in memory and returns a reference to the newly created object.

```
Employee  e; // Declaration
e = new Employee (); //Instantiation
```

- The reference 'e' is pointing to the Employee object in memory.
- The new operator allocates memory for the object.
- We can declare the reference e and instantiate the Employee object in a single statement:

```
Employee e = new Employee ();
//Declaration + Instantiation
```

Create Objects - within same Class

```
public class Employee
{
        //field or data member or instance variables
        int id;
        String name;

        public static void main(String args[])
        {
                Employee emp=new Employee();
                //creating an object of Employee

                System.out.println(emp.id);
                //accessing member through reference variable

                System.out.println(emp.name);
                 //accessing member through reference variable
        }
}
```

Create Objects outside the Class (Driver Class)

```
public class NewEmployee
{
        int id;
        String name;
 }

public class TestEmployee
{
    public static void main(String args[])
   {
        NewEmployee emp=new NewEmployee();

        System.out.println(emp.id);
        System.out.println(emp.name);
    }
}
```

Initializing Objects

- There are 3 ways to initialize object in java.
  - By reference variable
  - By method
  - By constructor

Initialization through reference

- Initializing object simply means storing data into object.

```
public class Employee
{
        int id;
        String name;
}
public class TestEmployee
{
    public static void main(String args[])
   {
        Employee emp=new Employee();
        emp.id=101;
        emp.name="Nimal";
        System.out.println("Employee id :"+emp.id+" ,Employee name :"+emp.name);
    }
}
```

Initialization through method

- Use a method to initialize objects and access objects values.

```
public class Student
{
        String name;
        int id;

        public void insertRecord(String s, int i)
        {
                name=s;
                id=i;
        }
        public void displayInformation()
        {
            System.out.println("Student name :" +name+" ,Student id :"+id);
        }
}
```

Initialization through method

```
class TestStudent
{
        public static void main(String args[])
        {
                Student stu1=new Student();
                Student stu2=new Student();

                stu1.insertRecord(111,“Saman");
                stu2.insertRecord(222,“Amal");

                stu1.displayInformation();
                stu2.displayInformation();
        }
}
```

Understanding Constructors

- In Java, constructor is a block of codes similar to a method.
- A constructor is called when a new instance of an object is created.
  - In fact, it's the new keyword that calls the constructor.
- After creating the object, you can't call the constructor again.
- It is a special type of method which is used to initialize the object.

Understanding Constructors

- When a constructor is called Every time, an object is created using new() keyword, at least one constructor is called.
  - It is called a default constructor.
- Why – the name ?
  - It is called constructor because it constructs the values at the time of object creation.
- It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

Creating Constructors

- Rules for creating java constructor
  - **Constructor name must be same as its class name**
  - **Constructor must have no explicit return type**

- Types of java constructors
  - **Default constructor (no-arg constructor)**
  - **Parameterized constructor**

Initialization through constructor

- **Use a constructor to initialize objects.**
- Constructors are used to initialize the instance variables of a given class.
- **They have the same name as that of their class**.
- **They have no return type** because they implicitly return an object of their class.

```
Employee emp = new Employee();
```

- Here, the default constructor Employee() is being invoked to initialize emp.
- **Default constructor takes no parameters**.
- **Default constructor initializes all instance variables to zero or null**.

Example

```
public class Employee
{
        private String Name;
        private int  Age;
        private char Gender;
}
```

- Suggest the Constructors
  - Default ?
  - Parameterized ?

Example – Default Constructor

```
public class Employee
{
        private String Name;
        private int  Age;
        private char Gender;

        Employee()
        {
                System.out.println("Default constructor executed…");
                System.out.println("Name : "+Name+" ,Age : "+Age+" ,Gender :"+Gender);
        }
}
```

Example – Parameterized Constructors

```
public class Employee
{
        private String Name;
        private int  Age;
        private char Gender;

        Employee( String n, int a, Char g )
        {
                Name = n;
                 Age = a;
                Gender = g;
                System.out.println("Parametarized constructer executed…");
                System.out.println("Name : "+Name+" ,Age : "+Age+" ,Gender :"+Gender);
        }
}
```

Homework - Try It Out

```
class Account
{
        int a,b;
        public void setData(int a, int b)
        {
                a=a;
                b=b;
        }
        public void showData(){
                System.out.println("Value of A=" +a);
                System.out.println("Value of B=" +b);
        }
        public static void main(String[] args)
        {
                Account myAccount= new Account();
                 myAccount.setData(2,3);
                 myAccount.showData();
        }
}
```

Summary

- Object Oriented Programming - Concepts
- Understanding Objects
- Understanding Classes
- Understanding Fields
- Understanding Methods
- JAVA - Access Modifiers
- Creating Objects
- Initializing Objects
  - By reference variable
  - By method
  - By constructor
- Understanding Constructors
  - Default
  - Parameterized

References

- https://docs.oracle.com/javase/tutorial/java/javaOO/index.html
- https://youtu.be/wuRt0AEE_m8
- How To Program (Early Objects)
  - By H .Deitel and P. Deitel
- Headfirst Java
  - By Kathy Sierra and Bert Bates

Questions ???

Thank You