



ሰኔ, 2013 ዓ.ም

# Object Oriented Programming



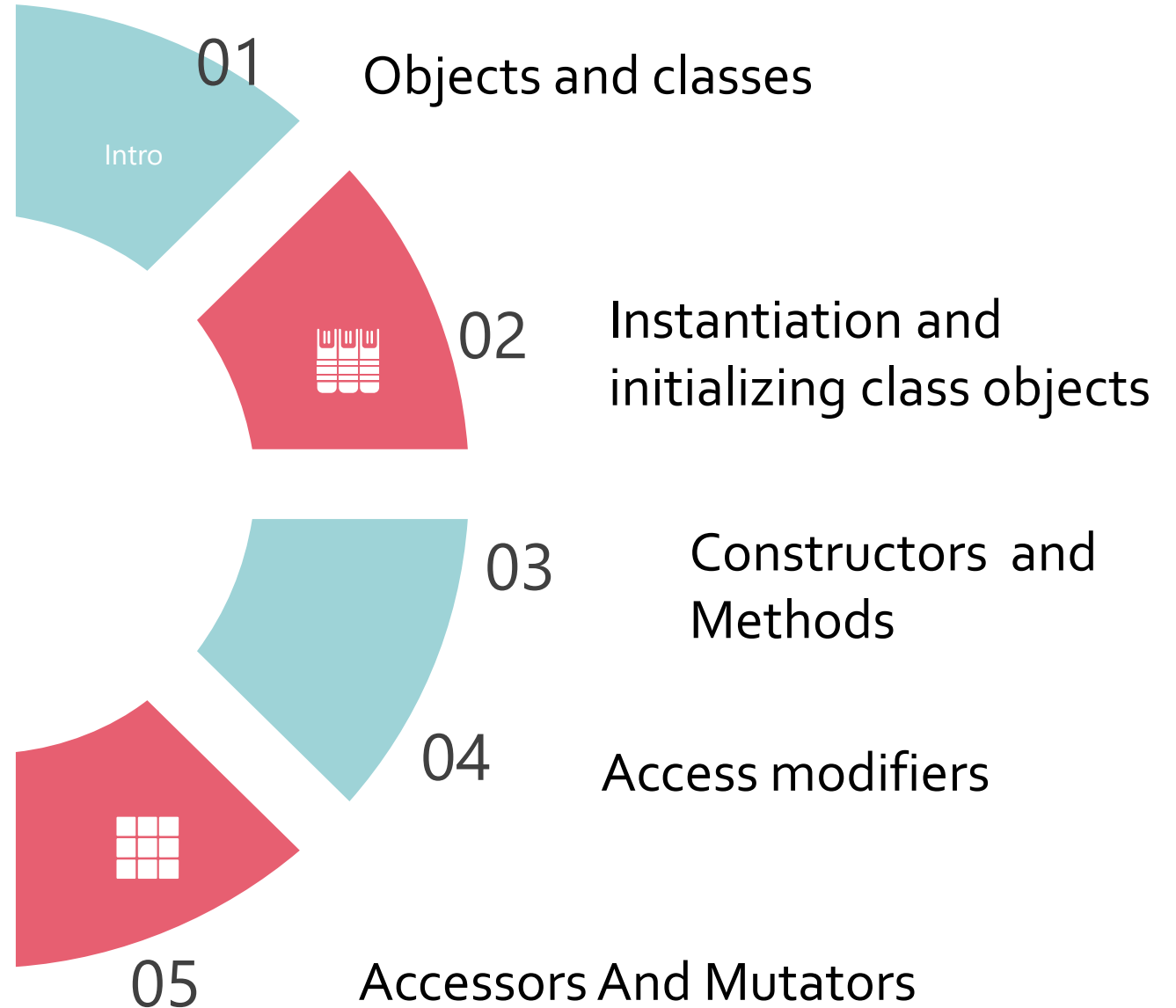
## Chapter Two

Wollo University, Kombolcha Institute of Technology.  
College of Informatics.  
Department of Information System.

By Daniel G.

# Contents

Object Oriented Programming  
chapter 2 will include this topics.



# Basics questions In this chapter

- 1. What is class?*
- 2. What is object?*
- 3. How to Instantiation and initializing class objects ?*
- 4. What is the difference between Constructors and Methods?*
- 5. What are Access modifiers in OOP?*
- 6. What are Accessors And Mutators?*
7. Differentiate static and instance members or variables in java ?



# What is Class and Object?

ሰኔ, 2013 ዓ.ም

- ❖ Everything in **Java** is associated with **classes** and **objects**, along with its **attributes** and **methods**.
- ❖ For example: in real life, a **car** is an **object**. The car has **attributes**, such as **weight** and **color**, and **methods**, such as **drive** and **brake**.

- ❖ A **Class** is like an object constructor
- ❖ A **class** is a group of **objects** which have common properties. It is a **template** or **blueprint** from which objects are created. It is a **logical** entity. It can't be physical.
- ❖ An **object** is the **instance** of a class

# Create class and object

ሰኔ, 2013 ዓ.ም

❖ To create a class use the key word **class**

```
public class Dog {  
    String breed;  
    int age;  
    String color;  
    void barking() {  
  
    }  
    void hungry() {  
  
    }  
    void sleeping() {  
  
    }  
}
```

- ❖ To create an object use the key word **new**
- ❖ The **new** keyword is used to allocate memory at runtime.

```
public class Dog {  
    String breed;  
    int age;  
    String color;  
    void barking() {  
    }  
    void hungry() {  
    }  
  
    public static void main(String []args) {  
        Dog mydog = new Dog();  
    }  
}
```

- ❖ The **new** keyword is a **Java** operator that creates the **object**, this is also known as **instantiating** a class.
- ❖ Initializing an object means storing data into the object
- ❖ There are 3 ways to initialize object in Java.
  1. By **reference** variable
  2. By **method**
  3. By **constructor**



# Example for **instantiating** class objects

ሰኔ, 2013 ዓ.ም

```
public class Dog {
    String breed;
    int age;
    String color;
    void barking() {
    }
    void hungry() {
    }
}

public static void main(String []args) {
    Dog mydog = new Dog(); //instantiating mydog object
}
```

# Example for **initializing** class objects

ሰኔ, 2013 ዓ.ም

```
public class Dog {  
    int age;  
    String color;  
}  
  
public class Bulldog{  
    void hungry() {  
    }  
}  
  
public static void main(String []args) {  
    Dog mydog = new Dog();  
    mydog.age =12; // initializing class objects by reference  
    System.out.print(mydog.age) ;  
}  
}
```

```
public class Dog {  
    int age;  
    String name;  
    void insert(int a, string c) {  
        age = a;  
        name= c;  
    }  
    public static void main(String []args) {  
        Dog mydog = new Dog();  
        mydog.insert(12, "tomy"); // initializing class objects by method  
    }  
}
```

```
public class Dog {  
    int age;  
    String color;  
    void Dog(int a, string c) {  
        age = a;  
        color = c;  
    }  
    public static void main(String []args) {  
        Dog mydog = new Dog(12, "tomy"); // initializing class objects by constructor  
    }  
}
```

- ❖ **Constructors** are used to initialize the object's state.
- ❖ Like methods, a constructor also contains **collection of statements(i.e. instructions)** that are executed at time of Object creation.
- ❖ Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members** of the same class.

```
public class Taxi
{
    public bool IsInitialized;
    public Taxi()//this is a constructor
    {
        IsInitialized = true;
    }
    Public static void Main()
    {
        Taxi t = new Taxi();
        console.WriteLine(t.IsInitialized);
    }
}
```

- ❖ A **method** is a collection of statements that perform some specific task and return the result to the caller.
- ❖ A **method** can perform some specific task without returning anything.  
Methods allow us to reuse the code without retyping the code

```
Public class Addition {  
  
    int sum = 0;  
    public int addTwoInt(int a, int b)  
    {  
        sum = a + b;  
        return sum;  
    }  
    public static void main(String[] args)  
    {  
        Addition add = new Addition();//instance of Addition class  
        System.out.println(add.addTwoInt(1, 2));// Callingmethod  
    }  
}
```



# The difference **methods** and **constructor**

ሰኔ, 2013 ዓ.ም

Constructors	Methods
A <b>Constructor</b> is a block of code that initializes a newly created object.	A <b>Method</b> is a collection of statements which returns a value upon its execution.
A <b>Constructor</b> can be used to initialize an object.	A <b>Method</b> consists of Java code to be executed.
A <b>Constructor</b> is invoked implicitly by the system.	A <b>Method</b> is invoked by the programmer.
A <b>Constructor</b> is invoked when a object is created using the keyword <b>new</b> .	A <b>Method</b> is invoked through method calls.

Constructors	Methods
A <b>Constructor</b> doesn't have a return type.	A <b>Method</b> must have a return type.
A <b>Constructor</b> initializes a object that doesn't exist.	A <b>Method</b> does operations on an already created object.
A <b>Constructor's</b> name must be same as the name of the class.	A <b>Method's</b> name can be anything.
A class can have many Constructors but must not have the same parameters.	A class can have many methods but must not have the same parameters.
A <b>Constructor</b> cannot be inherited by subclasses.	A <b>Method</b> can be inherited by subclasses.

- ❖ Access modifiers are keywords that help set the **visibility** and **accessibility** of a class, its member variables, constructors, and methods
- ❖ Access modifier determines whether a field or method in a class, can be used or invoked by another method in another class or subclass.

❖ . There are 4 modifiers in java

1. Private
2. Protected
3. No modifier or default or friendly access
4. Public

- ❖ The **two levels** are **class level** access modifiers and **member level** access modifiers.
- ❖ In **class level** access modifiers(java classes only) Only two access modifiers is allowed, **public** and **no modifier**
  - ✓ If a class is '**public**', then it can be **accessed** from **anywhere**.
  - ✓ If a class has '**no modifier**', then it can **only** be **accessed** from '**same package**'.

- ❖ In **Member level** access modifiers (java variables and java methods)
  - ✓ All the four **public**, **private**, **protected** and **no modifier** is allowed.

## 1. *Public* access modifier

- ❖ This is the **least restrictive** access modifier .
- ❖ **Classes, fields, methods, constructors** and **interfaces** defined using the **public** access modifier are accessible or visible across all packages, from derived to unrelated classes

	Same package	Separate package
Derived classes	✓	✓
Unrelated classes	✓	✓

## 2. *Private* access modifier

- ❖ The **private** (most restrictive) fields or methods can be accessed only by the **declaring class**.
- ❖ Fields, methods or constructors declared **private** are strictly controlled, which means they **cannot be accesses** by anywhere outside the enclosing class.

	Same package	Separate package
Derived classes	✗	✗
Unrelated classes	✗	✗



### 3. *Protected access modifier*

- ❖ The members of a class defined using the protected access modifier are accessible to
  - ✓ Classes, fields, methods, constructors and interfaces defined in the same package
  - ✓ All derived classes, even if they're defined in separate packages
- ❖ Fields, methods and constructors declared protected in a superclass can be accessed only by subclasses in other packages

	Same package	Separate package
Derived classes	✓	✓
Unrelated classes	✓	✗

## 4. No modifier (default access modifier)

- ❖ The members of a class defined without using any **explicit access modifier** are defined with package accessibility (also called **default accessibility**).
- ❖ The members with package access are only accessible to classes, field, method or constructor and interfaces defined in the same package.

	Same package	Separate package
Derived classes	✓	✗
Unrelated classes	✓	✗

# Conti.....

## Example

ሰኔ, 2013 ዓ.ም

**package p1;**

```
public class C1 {  
    public int x;  
    int y;  
    private int z;  
  
    public void m1() {  
    }  
    void m2() {  
    }  
    private void m3() {  
    }  
}
```

```
public class C2 {  
    void aMethod() {  
        C1 o = new C1();  
        can access o.x;  
        can access o.y;  
        cannot access o.z;  
  
        can invoke o.m1();  
        can invoke o.m2();  
        cannot invoke o.m3();  
    }  
}
```

**package p2;**

```
public class C3 {  
    void aMethod() {  
        C1 o = new C1();  
        can access o.x;  
        cannot access o.y;  
        cannot access o.z;  
  
        can invoke o.m1();  
        cannot invoke o.m2();  
        cannot invoke o.m3();  
    }  
}
```

**package p1;**

```
class C1 {  
    ...  
}
```

```
public class C2 {  
    can access C1  
}
```

**package p2;**

```
public class C3 {  
    cannot access C1;  
    can access C2;  
}
```

- ❖ In Java **accessors** are used to **get** the value of a private field and **mutators** are used to **set** the value of a private field.
- ❖ **Accessors** are also known as **getters** and **mutators** are also known as **setters**.
- ❖ If we have declared the variables as **private** then they would not be accessible by all so we need to use **getter** and **setter** methods.

- ❖ An **Accessor** method is commonly known as a **get** method or simply a **getter**. A property of the object is returned by the accessor method. They are declared as **public**. A naming scheme is followed by **accessors**, in other words they add a word to get in the start of the method name. They are used to return the value of a **private** field. The same data type is returned by these methods depending on their private field.

## Syntax

```
public int getNumber()  
{  
    return Number;  
}
```

## Example

```
public class Employee {  
    private int number;  
    public int getNumber() {  
        return number;  
    }  
    public void setNumber(int newNumber) {  
        number = newNumber;  
    }  
}
```

- ❖ A **Mutator** method is commonly known as a **set** method or simply a **setter**. A **Mutator** method mutates things, in other words change things. It shows us the principle of **encapsulation**. They are also known as **modifiers**. They are declared as **public**. **Mutator** methods do not have any return type and they also accept a parameter of the same data type depending on their **private** field. After that it is used to set the value of the **private** field.

## Syntax

```
public void set Age(int Age) {  
    this.Age = Age;  
}
```

## Example

```
public class Cat {  
    private int Age;  
    public int getAge() {  
        return this.Age;  
    }  
    public void set Age(int Age) {  
        this.Age = Age;  
    }  
}
```



# Calling and returning methods

ሰኔ, 2013 ዓ.ም

- ❖ Formal **parameters** allow you to pass a value into a method. A method can also pass a value out when it is finished.
- ❖ If a **method** is a **void method**, like most of the methods we have seen so far, it does not return a value when called. But some methods **return** a value when called.
- ❖ **N.B** Calling Methods that Return Values when function is not void type

## Example

```
public class Age {  
    private int Age;  
    public int myage(int a) {  
        a =this.Age;  
        return a;  
    }  
    public static void main() {  
        Age my = new Age;  
        System.out.print(my.myage(12) );  
    }  
}
```

# Static and instance members

ሰኔ, 2013 ዓ.ም

Instance variables	Static (class) variables
Instance variables are declared in a class, but outside a method, constructor or any block.	Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.	Static variables are created when the program starts and destroyed when the program stops.

Instance variables	Static (class) variables
<p>Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. <code>ObjectReference.VariableName</code>.</p>	<p>Static variables can be accessed by calling with the class name <code>ClassName.VariableName</code>.</p>
<p>Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.</p>	<p>There would only be one copy of each class variable per class, regardless of how many objects are created from it.</p>

## Example

```
public class Variable {  
    int myVariable; //instance variable  
    static int data = 30; //static variable  
    public static void main(String args[]) {  
        Variable obj = new Variable ();  
        System.out.println("instance variable: "+obj.myVariable);  
        System.out.println("static variable: "+Variable.data);  
    }  
}
```

# Accepting input from the user

ሰኔ, 2013 ዓ.ም

```
package pro;
import java.util.Scanner;
public class Pro {
    public static void main(String[] args) {

        String name;
        int length, width, area;
        Scanner console = new Scanner(System.in);
        System.out.println("enter your name ");
        name = console.nextLine();
        System.out.print("Enter length ");
        length = console.nextInt();
        System.out.print("Enter width ");
        width = console.nextInt(); area = length * width;
        System.out.println("The area of rectangle is " + area);
    }
}
```

7/5/2021

By Daniel G.

38

ሰኔ, 2013 ዓ.ም

# THANK YOU

**Any questions?**  
Feel free !