Object Oriented Development

ICT2123

Classes, Objects, Methods and Constructors

Chanduni Gamage
Department of ICT
Faculty of Technology
University of Ruhuna

Lecture 3

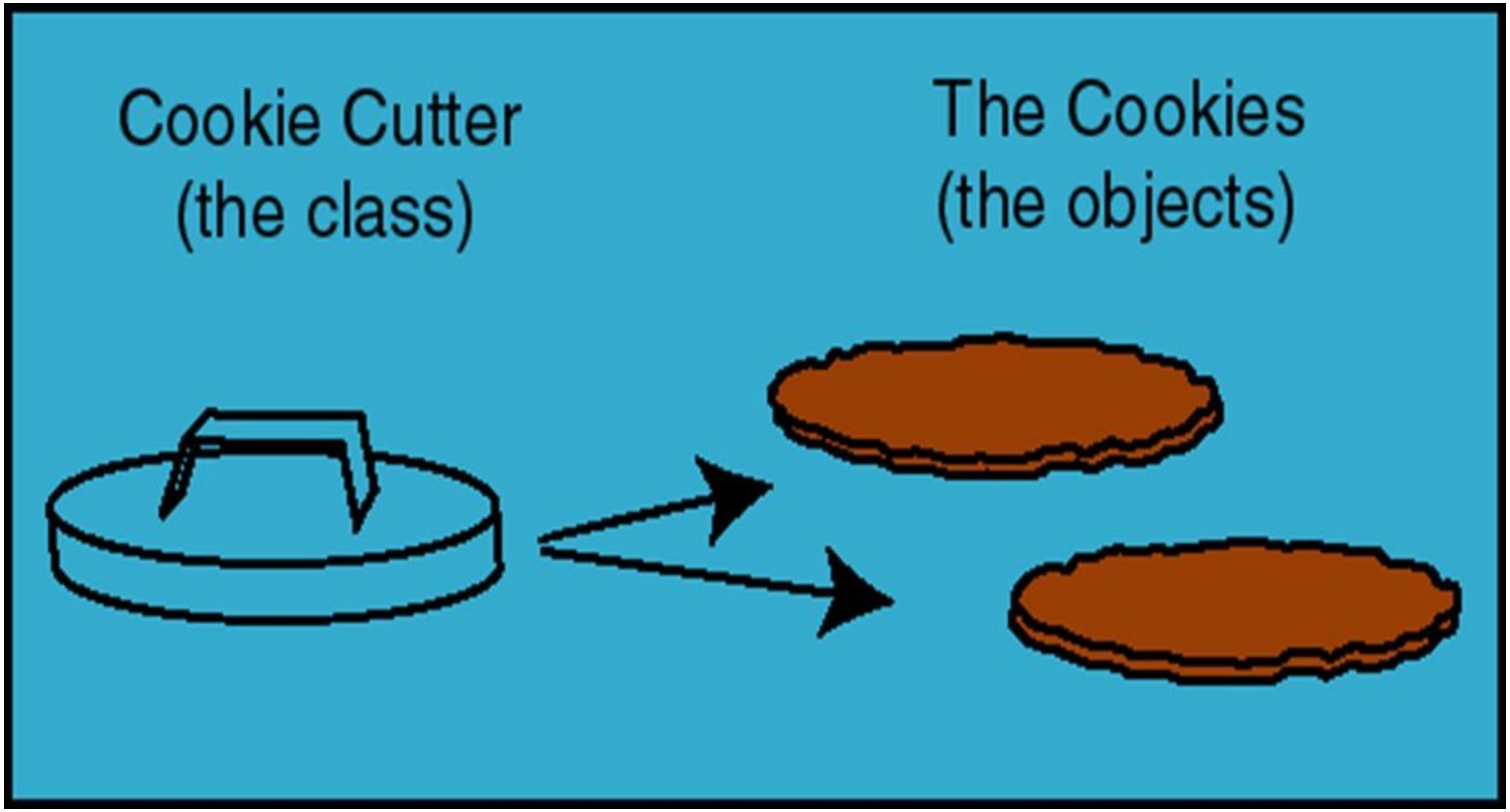
What we discuss Today ……..

- *Objects*
- *Classes*
- *Declaration, Instantiation and Initialization of Objects*
- *Constructors*
- *Static methods*
- *Anonymous objects*
- *Java Garbage Collection*

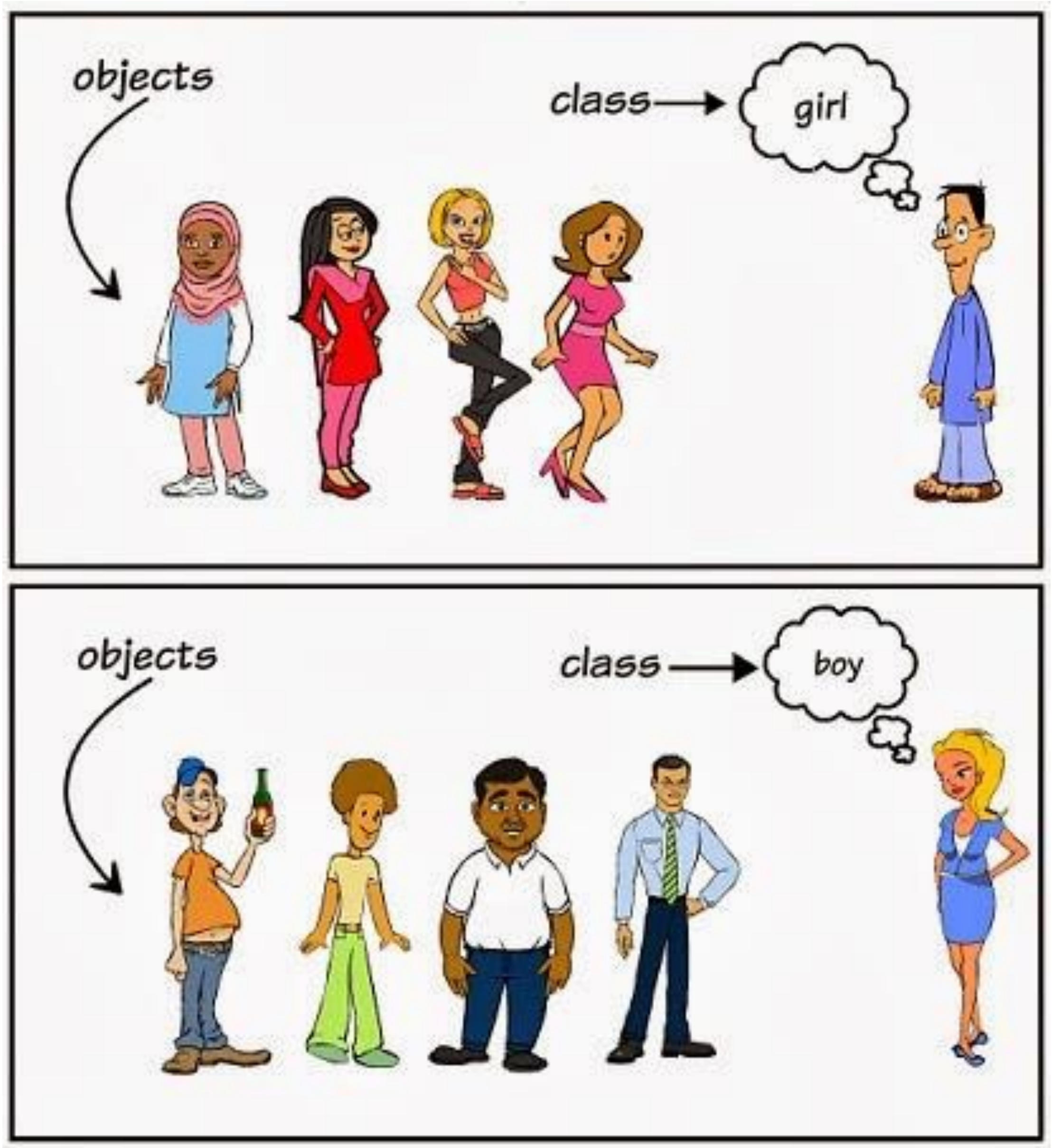
Object Oriented Concepts

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

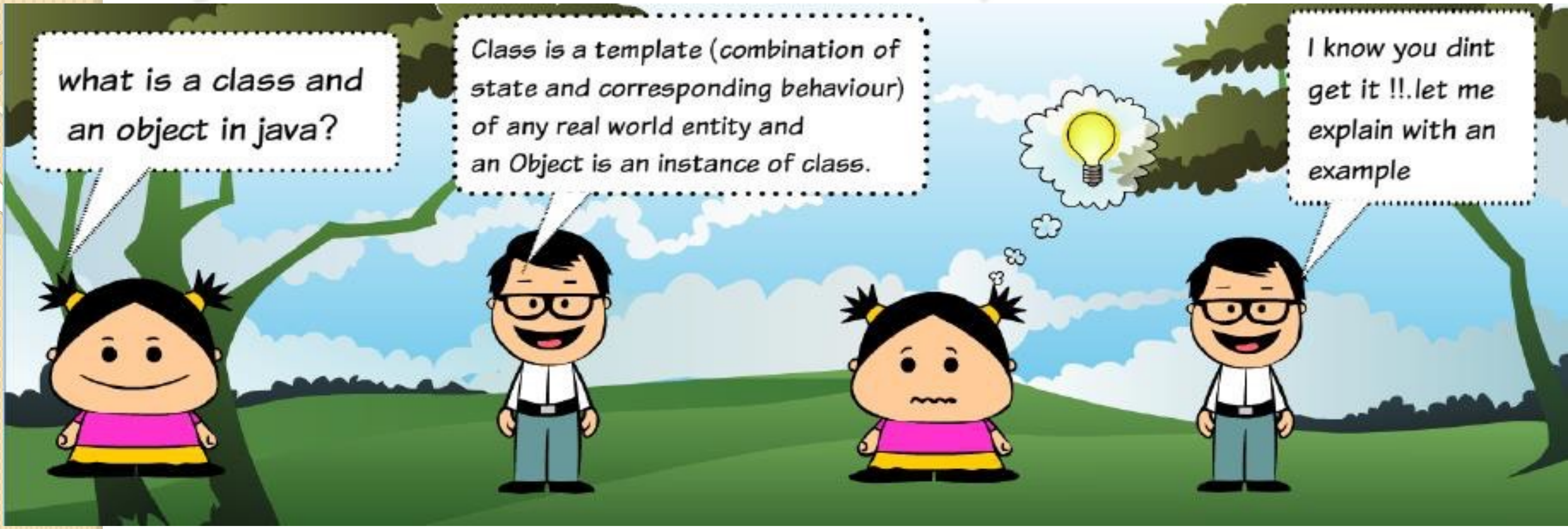
JAVA Class and Object



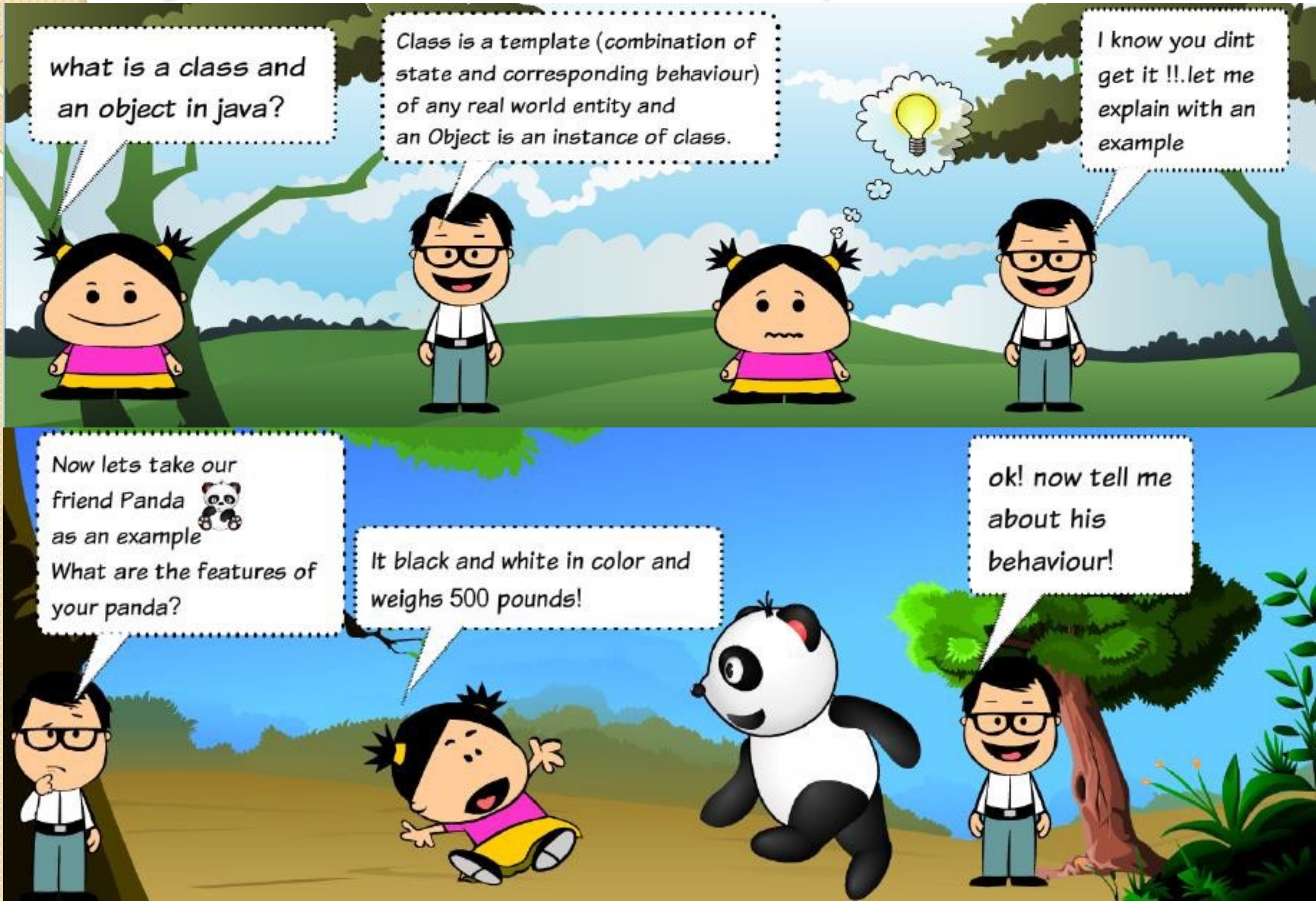
JAVA Class and Object



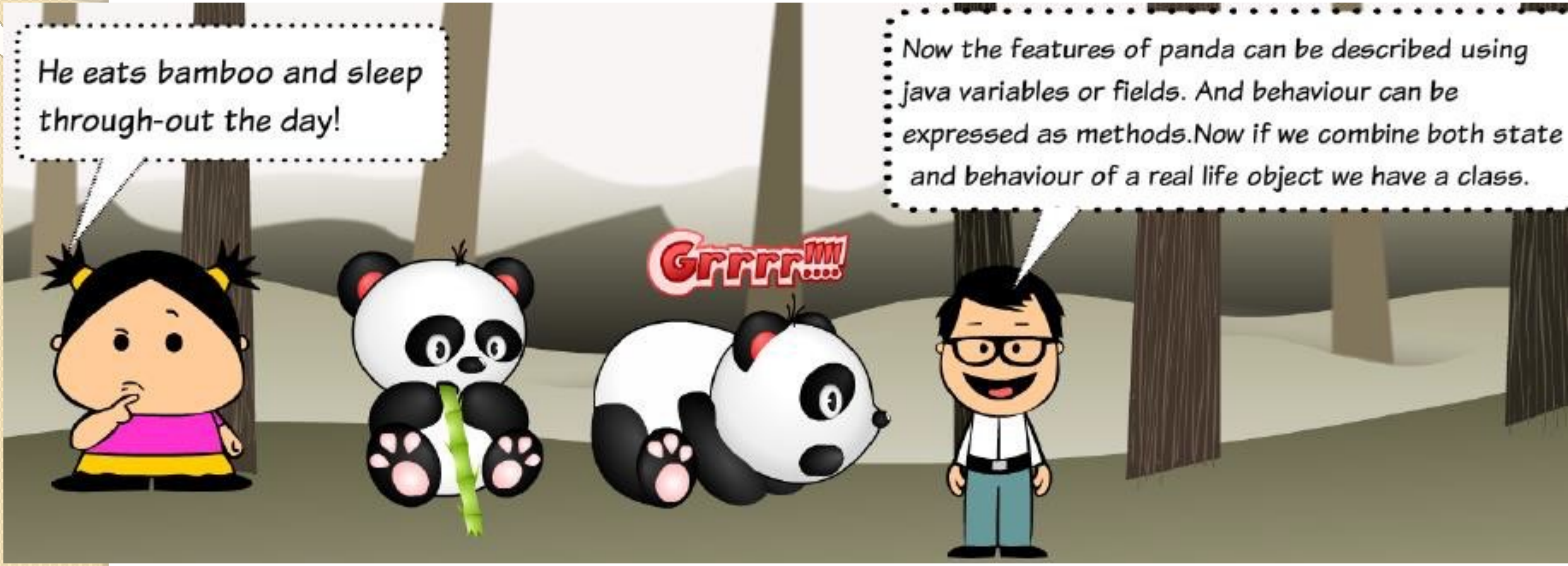
JAVA Class and Object


what is a class and an object in java?
Class is a template (combination of state and corresponding behaviour) of any real world entity and an Object is an instance of class.
I know you dint get it !!.let me explain with an example

JAVA Class and Object


what is a class and an object in java?
Class is a template (combination of state and corresponding behaviour) of any real world entity and an Object is an instance of class.
I know you dint get it !!.let me explain with an example
Now lets take our friend Panda as an example What are the features of your panda?
It black and white in color and weighs 500 pounds!
ok! now tell me about his behaviour!

JAVA Class and Object


He eats bamboo and sleep through-out the day!
Grrrr!!!!
Now the features of panda can be described using java variables or fields. And behaviour can be expressed as methods.Now if we combine both state and behaviour of a real life object we have a class.

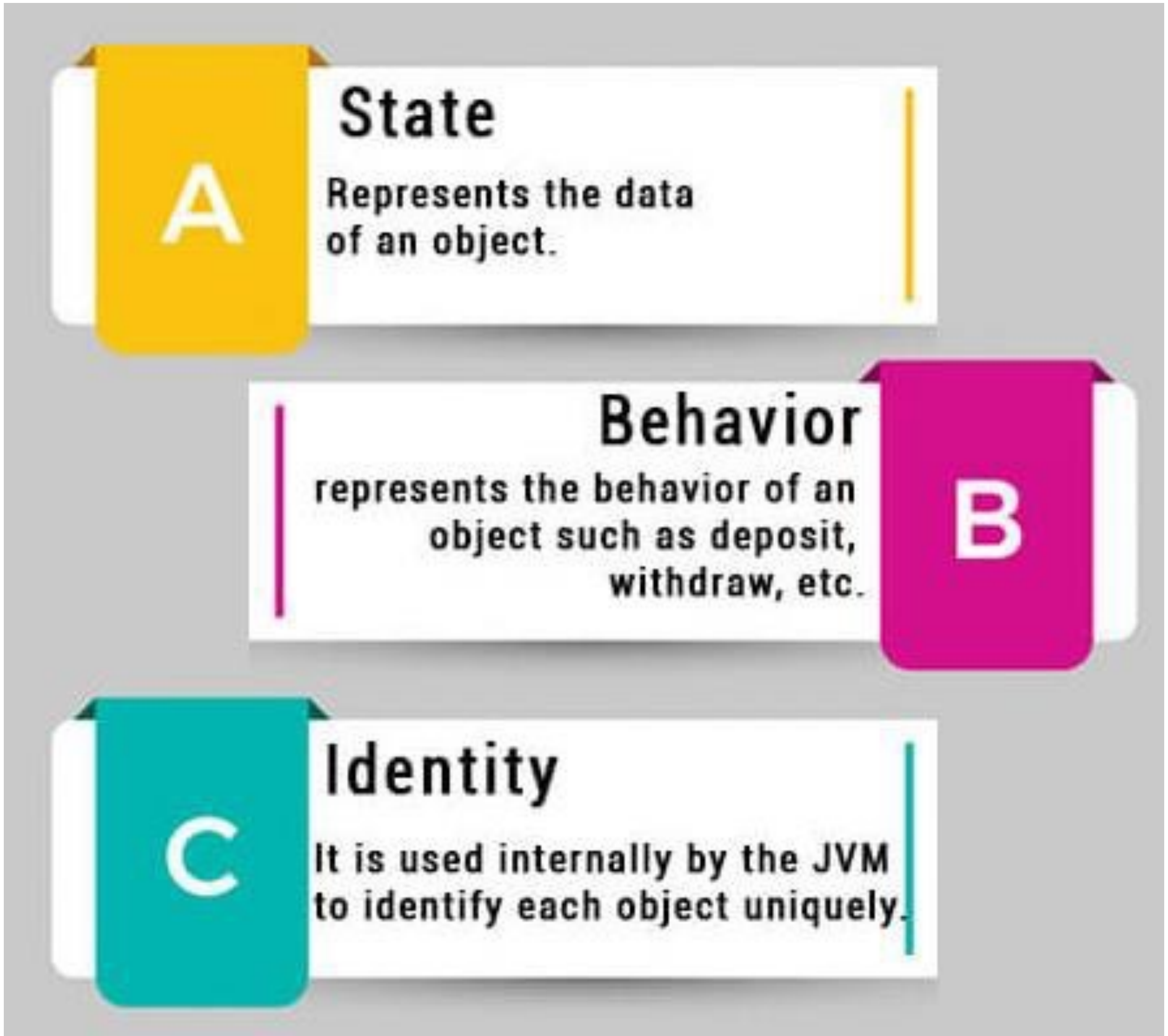
What Is an Object?

- An object is a (software) bundle of related **state** and **behavior**.
- It can be physical or logical (tangible and intangible).
  - Examples ???
- Software objects are often used to model the real-world objects that you find in everyday life.

What Is an Object?

- Real-world objects share two characteristics:
  - *state*
  - *behavior*
- Software objects are conceptually similar to real-world objects:
  - consist of state and related behavior
  - An object stores its state in *fields* (variables in some programming languages) and
  - exposes its behavior through *methods* (functions in some programming languages).
  - Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

Characteristics of Object



A

State

Represents the data of an object.

B

Behavior

represents the behavior of an object such as deposit, withdraw, etc.

C

Identity

It is used internally by the JVM to identify each object uniquely.

What is a Class?

- In the real world, you'll often find many individual objects all the same kind.
- There may be thousands of motor bicycles in existence, consider about motor bicycles of the same make and model.
- Each motor bicycle was built from the same set of blueprints and therefore contains the same components.
- In object-oriented terms, we say that your motor bicycle is an instance of the class of objects known as bicycles.
- **A class is the blueprint from which individual objects are created.**

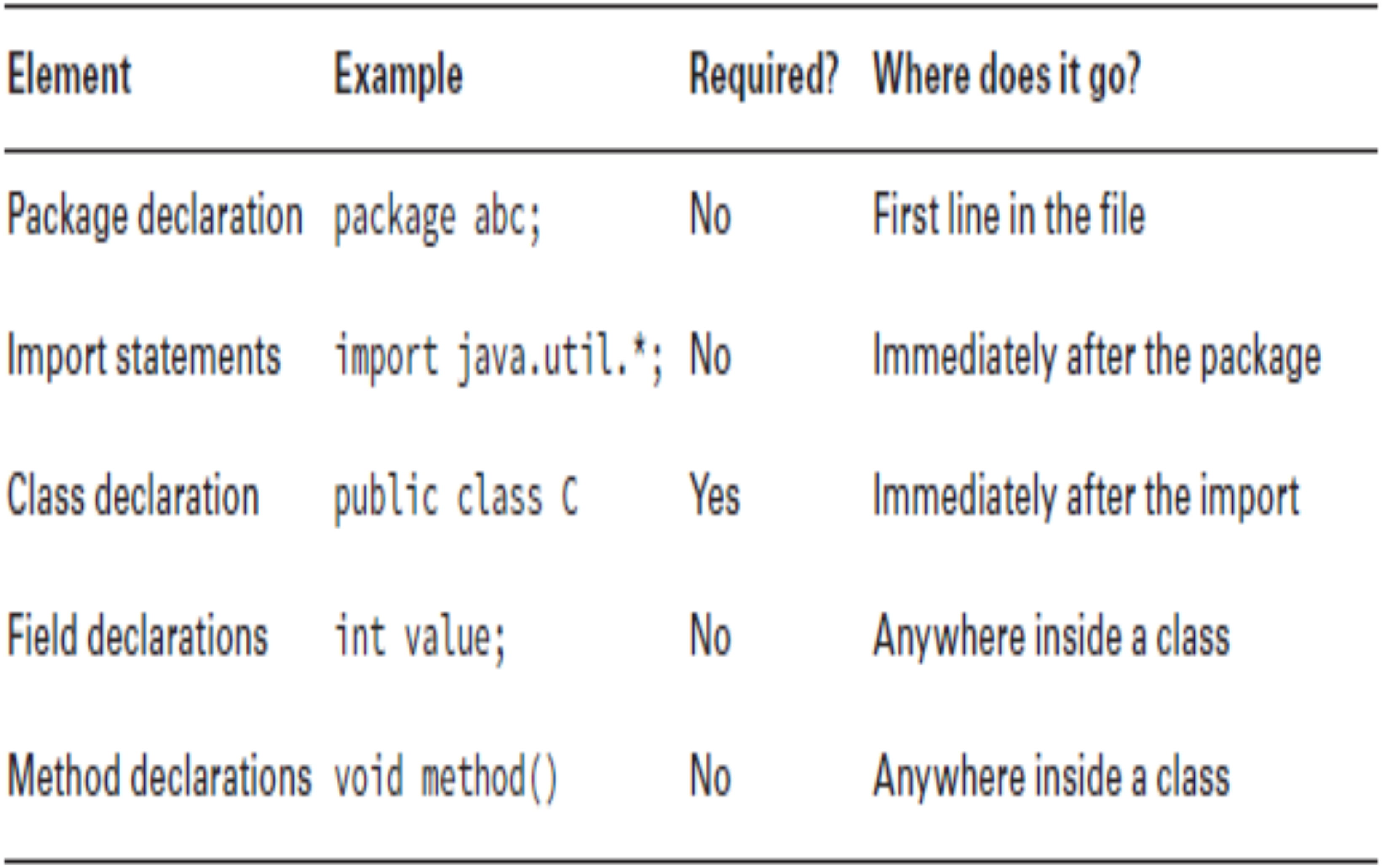
Class Definition

- **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.**

Syntax:

```
class <class_name>{
        field;
        method;
        constructor;
        blocks;
        nested classes and interface;
}
```

Elements of 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

Life Cycle of an Object

- In Java, it has seven states in Object lifecycle.
  - Created
  - In use
  - Invisible
  - Unreachable
  - Collected
  - Finalized
  - De-allocated

Life Cycle of an Object

- Creating an Object
  - **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.

Creating/Instantiating 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 Employee
{
            int id;
            String name;
  }

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

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

Initializing Objects

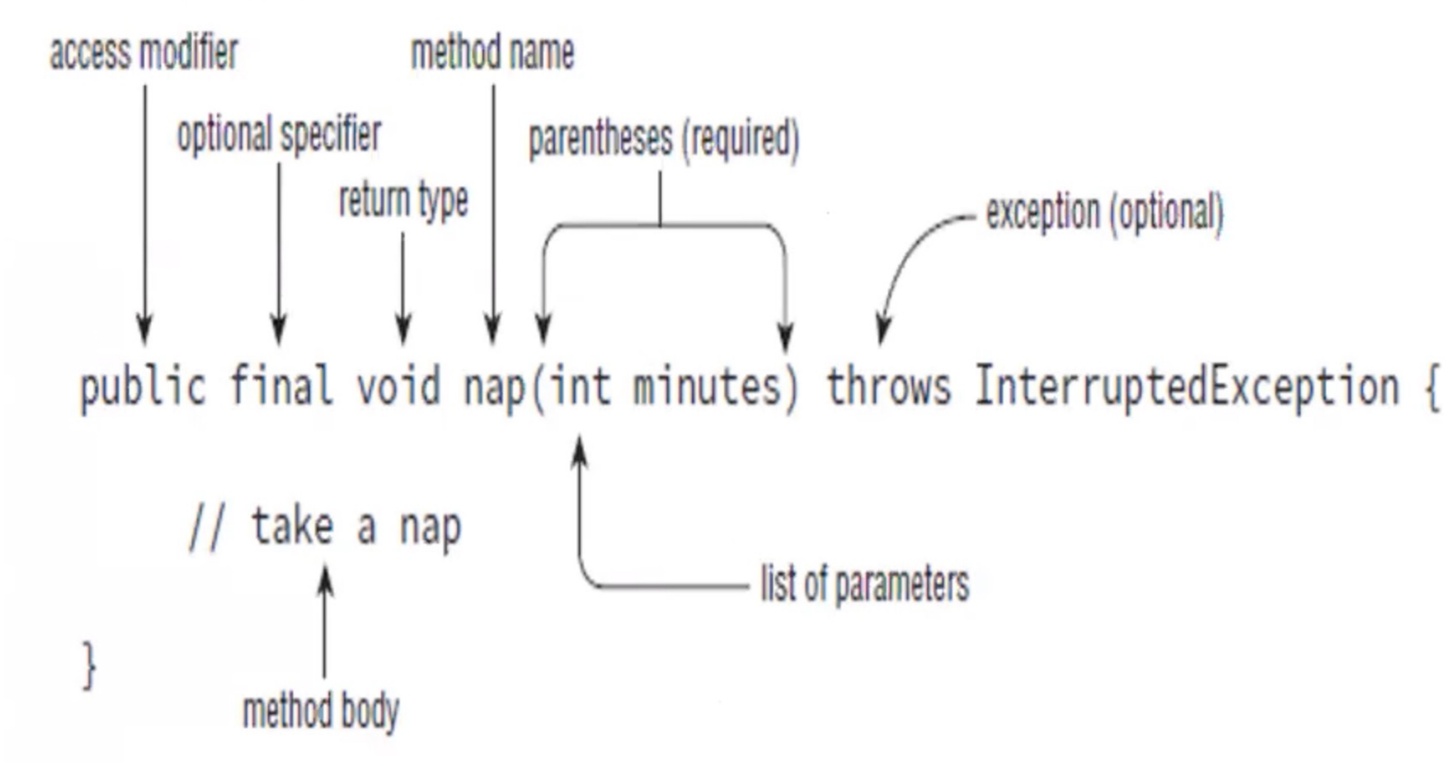
- There are 3 ways to initialize object in java.
  - By reference variable
  - By method
  - By constructor

Initialization through reference

- Initializing object simply means storing data into object.

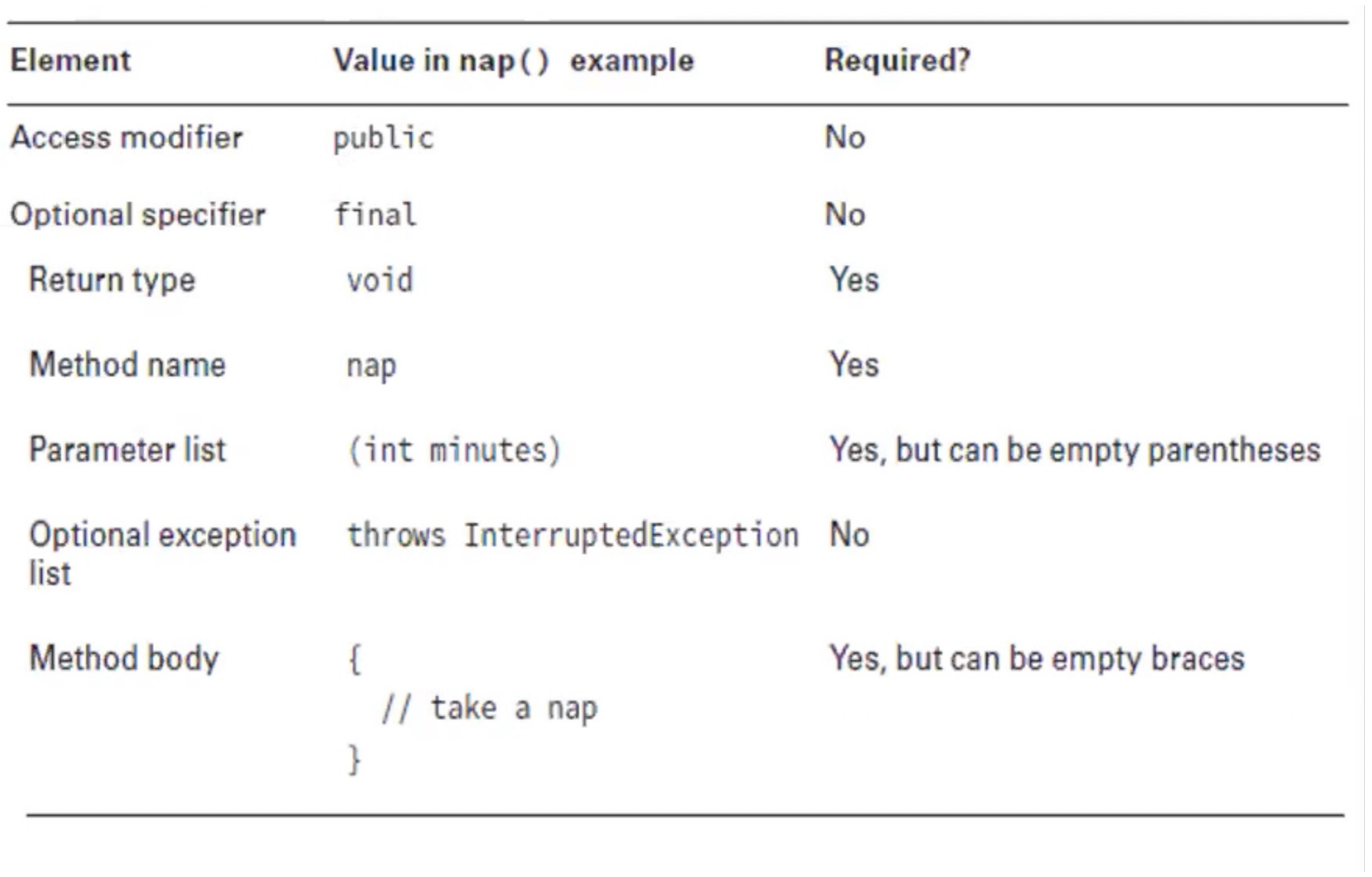
```
public class Employee
{
            int id;
            String name;
}
public class TestEmployee
{
     public static void main(String args[])
    {
            Employee emp=new Employee();
            emp.id=101;
            emp.name="Nimal";
            System.out.println("Employee id :"+emp.id+"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

Initialization through method



```
public final void nap(int minutes) throws InterruptedException {
    // take a nap
}
```

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

Initialization through method

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

```
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();
	}
}
```

Java Access Modifiers

Java offers four types 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**

  When no access modifier is specified for a class , method or data member. Accessible only within the same package.

Example

```
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();
        }
}
```

Example

- Why?
  - Both local and instance variables are same.
- Solution???
  - The “this” reference
  - Every object has a reference to itself represented by the “this” keyword
- Change code segment to

```
public void setData(int a, int b){
    this.a=a;
    this.b=b;
  }
```

- In the compilation time this will replace with myAccount. then left-hand side is instance variable, right hand side is local variable.

Constructors

- In Java, constructor is a block of codes similar to a method.
- It is called when an instance of object is created, and memory is allocated for the object.
- It is a special type of method which is used to initialize the object.
- When a constructor is called Everytime an object is created using new() keyword, at least one constructor is called. It is called a default constructor.
    - 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.

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;
}
```

Example

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

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

Example

```
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;
        }
}
```

More on Constructors

```
Employee( String n, int a, Char g )
        {
                this.Name =  n;
                this.Age =  a;
                this.Gender =  g;
        }
```

"this" keyword is optional

```
Employee( String Name, int Age, Char Gender )
        {
                this.Name =  Name;
                this.Age = Age;
                this.Gender =  Gender;
        }
```

"this" keyword is required

Anonymous objects

- Anonymous simply means nameless.
- An object which has no reference is known as anonymous object.
- It can be used at the time of object creation only.
- If you have to use an object only once, anonymous object is a good approach.

  Ex:

  ```
  new Calculation();//anonymous object
  ```

Anonymous objects

- Calling method through reference,

  Calculation c=new Calculation();

  c.fact(5);

- Calling method through anonymous object,

  ```
  new Calculation().fact(5);
  ```

How can an object be unreferenced?

- By nulling the reference
  Employee e=new Employee();
  e=null;

- By assigning a reference to another
  Employee e1=new Employee(); Employee e2=new Employee(); e1=e2;//now the first object referred by
  e1 is available for garbage
collection

- By annonymous object etc.
  new Employee();

Java Garbage Collection

- In java, garbage means unreferenced objects.
- Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.
- The Java runtime environment deletes objects when it determines that they are no longer being used.
- In java it is performed automatically. So, java provides better memory management.

Advantages of Garbage Collection

- It makes java memory efficient because garbage collector removes the unreferenced objects from heap memory.
- It is automatically done by the garbage collector(a part of JVM) so we don't need to make extra efforts.

Static methods

- We can also define static methods as global methods
- Static methods have several restrictions:
  - They can **only call other static methods**
  - They can **only access static data**
  - They **cannot access 'this' or 'super'** in any way
- One may also use a static block to initialize static variables
- As soon as class is loaded, all of the static statements will run

Example

```
public class UseStatic {
        static int a = 3;
        static int b;

        static void meth(int x)
       {
           System.out.println("x  =   " + x);
           System.out.println("a  =   " + a);
           System.out.println("b =    " + b);
        }

    static {
        System.out.println(" Static block initialized ");
        b = a *4;
    }

    public static void main(String[] args) {
              meth(42);
    }
}
```

Summary

- Objects
  - Characteristics
  - Benefits
- Classes
  - Definition
- Declaration, Instantiation and Initialization of Objects
  - By reference variable
  - By method
  - By constructor
- Constructors
  - Default Constructor
  - Parameterized Constructors
- Anonymous objects
- Java Garbage Collection
- Static methods

References

- How To Program (Early Objects)
  - 10th Edition
  - By H .Deitel and P.Deitel
- Head First Java
  - 2nd Edition
  - By Kathy Sierra and Bert Bates

Questions ???

Thank You