Implementing OOP in Java

Session 4

Objectives

- Object Orientation
- Method Overloading
- Encapsulation
- Access Modifiers
- Implement Constructors
- Implement Destructors in
- Explain the working of Garbage Collector
- Static Members
- This pointer

Object-orientation

- Method of designing and implementing software systems
- A technique for system modeling
- OO model consists of several interacting objects

What is a Model?

- A model is an abstraction of something
- Purpose is to understand the product before developing it
- Examples
  - Highway maps
  - Architectural models
  - Mechanical models

Object-Orientation - Advantages

- People think in terms of objects
- OO models map to reality
- Therefore, OO models are
  - easy to develop
  - easy to understand

What is an Object?

An object is

- Something tangible (Ali, Car)
- Something that can be apprehended intellectually (Time, Date)

Object

An object has

- State (attributes)
- Well-defined behavior (operations)
- Unique identity

Example – Ali is a Tangible Object

- State (attributes)
  - Name
  - Age
- behavior (operations)
  - Walks
  - Eats
- Identity
  - His name

Example – Car is a Tangible Object

- State (attributes)
  - Color
  - Model
- behavior (operations)
  - Accelerate
  - Start Car
  - Change Gear
- Identity
  - Its registration number

Example – Time is an Object Apprehended Intellectually

- State (attributes)
  - Hours
  - Minutes
  - Seconds
- behaviour (operations)
  - Set Hours
  - Set Minutes
  - Set Seconds
- Identity
  - Would have a unique ID in the model

Example – Date is an Object Apprehended Intellectually

- State (attributes)
  - Year - Day
  - Month
- behaviour (operations)
  - Set Year - Set Day
  - Set Month
- Identity
  - Would have a unique ID in the model

Data Abstraction

Characteristics of a Person

Name

Address

Age

Height

Hair color

Characteristics of a Customer of a car dealership

Name

Address

Data Abstraction (Contd...)

Attributes	**Actions**
Name of the Customer	Accept name of the customer
Address of the Customer	Accept address of the customer
Model of the car bought	Accept the model of the car purchased
Salesman who sold the car	Accept the salesman name who sold the car
	Generate the bill

Data Abstraction (Cont…)

- Data Abstraction is the process of identifying and grouping attributes and actions related to a particular entity as relevant to the application at hand
- Advantages
  - It focuses on the problem
  - It identifies the essential characteristics and actions
  - It helps to eliminate unnecessary detail

Method Overloading

- is defining different versions of a method in class, and the compiler will automatically select the most appropriate one based on the parameters supplied
- Method overloading is achieved by passing:
  - Different types of parameters
  - Different number of parameters
  - Different sequence of parameters

```
void display();      // Display methods
void display(int one, char two);
void display(int a, int b, int c);
void display(char one, int two);
```

Advantages

- Eliminates use of different method names for the same operation
- Helps to understand and debug code easily
- Maintaining code is easier

Overloading using different Data Types

```
int square(int);
float square(bool);
double square(char);
```

The compiler can distinguish between overloaded methods with same number of arguments provided their type is different

Overloading using Different Number/Sequence of Arguments

```
int square(int, char)

Int square(char,int)
int square(int,int, int)
```

```
int asq=square(3,'a')
int bsq=square(1,2,3)

int asq=square('a',3)
```

At compile time, compiler compares the types of actual arguments with the types of formal arguments of all methods called square

Construction

- The process of bringing an object into existence is called Construction
- A Constructor
  - Allocates memory
  - Initializes attributes, if any
  - Enables access to attributes and methods

Destruction

- The process of deleting an object is called Destruction
- A Destructor
  - Frees allocated space
  - Disables access to attributes and methods

Constructors

- They are special types of methods in a class.
- They are called every time an object is created.
- They are generally used for initialization.
- They have the same name as the class.
- They return no value.

Constructors

A constructor is a special method for automatic initialization of an object

General syntax for constructors is:

```
class username{
.
username() //constructor
        {
        ……
      …..//Code written
        }
};
```

Default Constructor in Java

Defines constructor

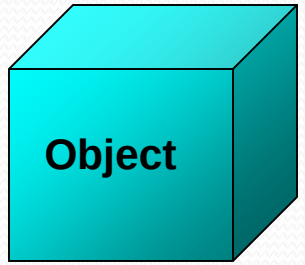
Default constructor not used

No

If we do not define any constructor then Java invokes default constructor for the class

Java

Constructor for the class



Default Constructor in Java

```
class Sdate {
       int month;
     int day;
     int year;
       Sdate()   //Constructor
       {
         month=11;
         day=27;
         year=1969;
       }
     public static void main(String args[])
     {
       Sdate S1,S2;
       S1=new Sdate();
       S2=new Sdate();
     }
 }
```

Parameterized Constructor

Constructors defined with parameters are known as parameterized constructor

```
class Sdate {
  int month;
  int day;
  int year;
  Sdate(int m,int d,int y){
            month=m;
            day=d;
            year=y;
           }
  public static void main(String args[]) {
       Sdate S1,S2;
       S1=new Sdate(11,27,1969);
       S2=new Sdate(3,3,1973);
}
```

Static Constructors

- They will be called only once before the first object is created.
- A static constructor is used to initialize any static data, or to perform a particular action that needs to be performed only once.
- They can be declared in the same way as a static method is declared.
- They cannot have any parameters.

```
…
static semester()
      {
      //Static Constructor Implementation
      }
…
```

Private Constructors

- Private constructors are used **to prevent creating instances of a class when there are no instance fields or methods**, such as the Math class, or when a method is called to obtain an instance of a class. If all the methods in the class are static, consider making the complete class static.

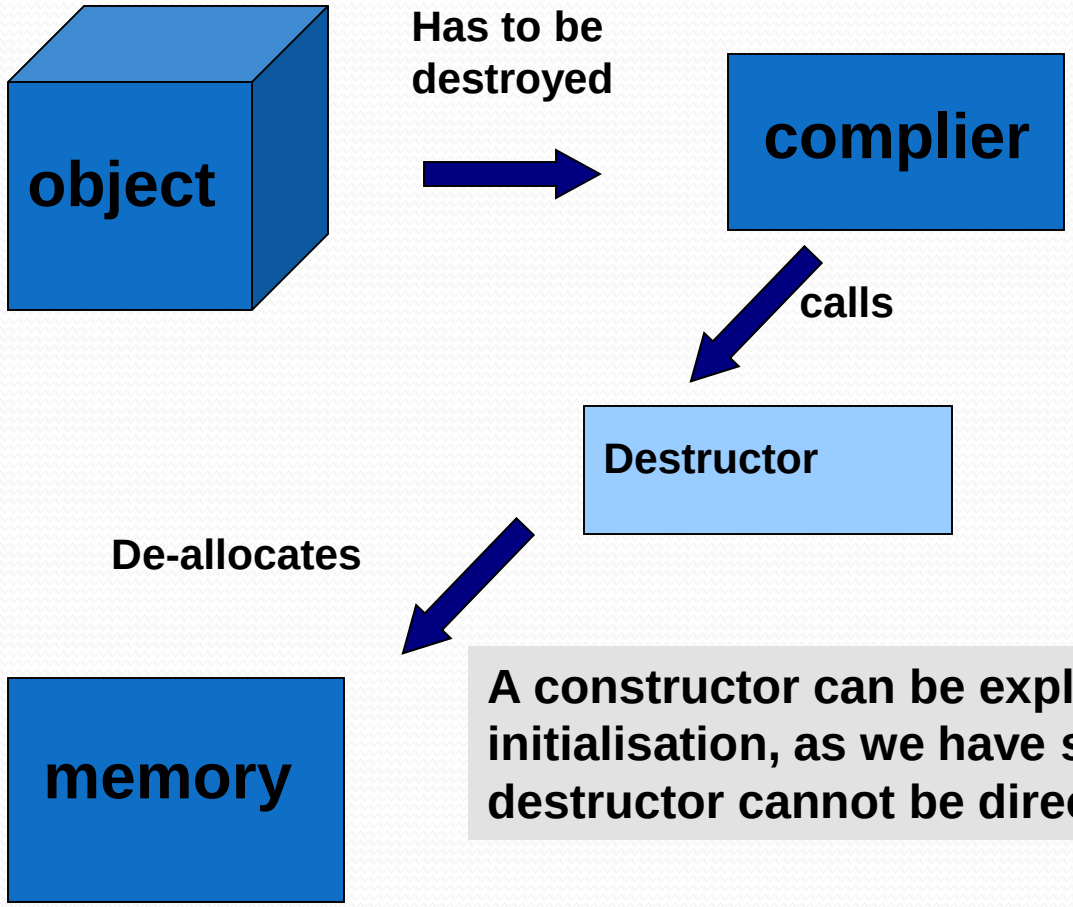
- The addition of the private access modifier does not make any difference to the accessibility of the constructor.

```
…
class AllStatic
{
private AllStatic()
      {
      //Private Constructor Implementation
      }
}
…
```

Destructors

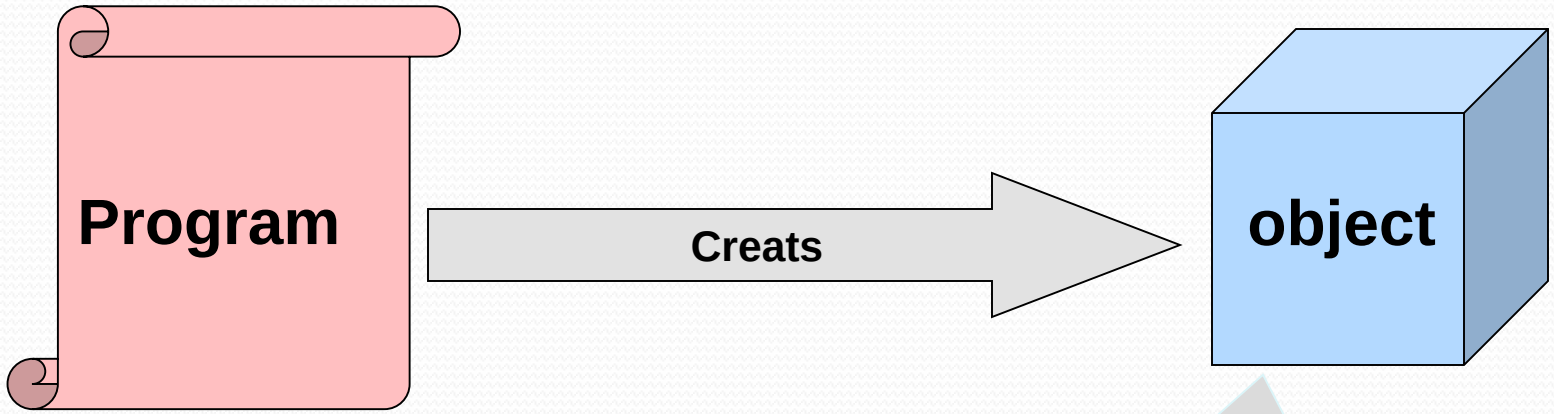
- They are called by **Garbage Collector** in Java
- Garbage Collector frees memory by destroying objects that are no longer required/referenced.

Destructors



A constructor can be explicitly called at the time of initialisation, as we have seen earlier, but a destructor cannot be directly called from the class.

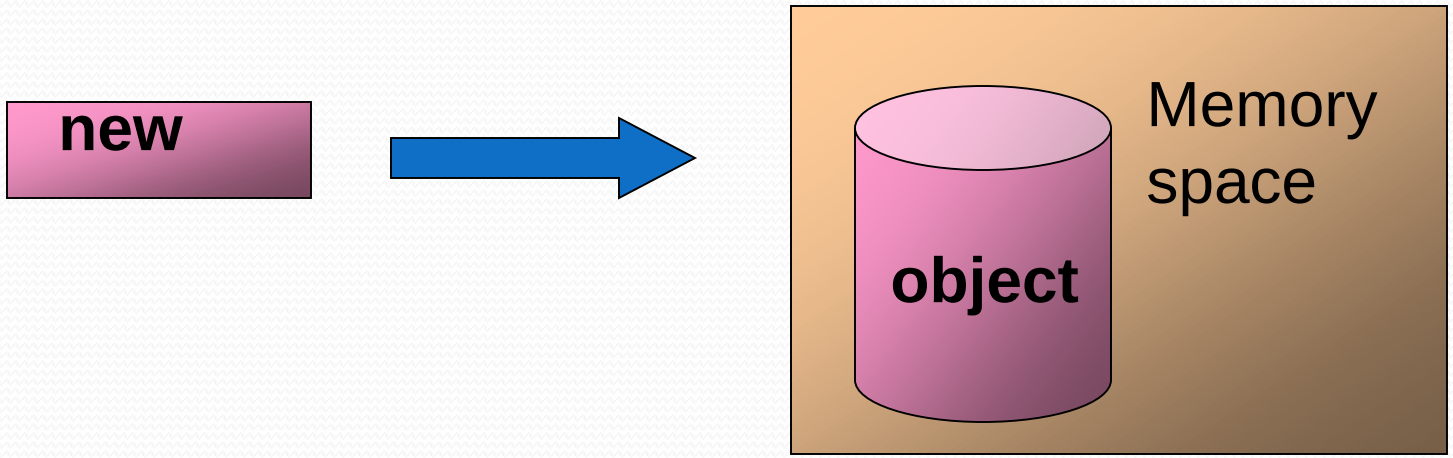
Memory allocation for an Object



It is useful to create a new object that will exist only as long as it is needed, otherwise huge memory will be occupied by all these unused objects.

According to the definition given in the Class.

Allocating Memory



int[] p = new int[3];

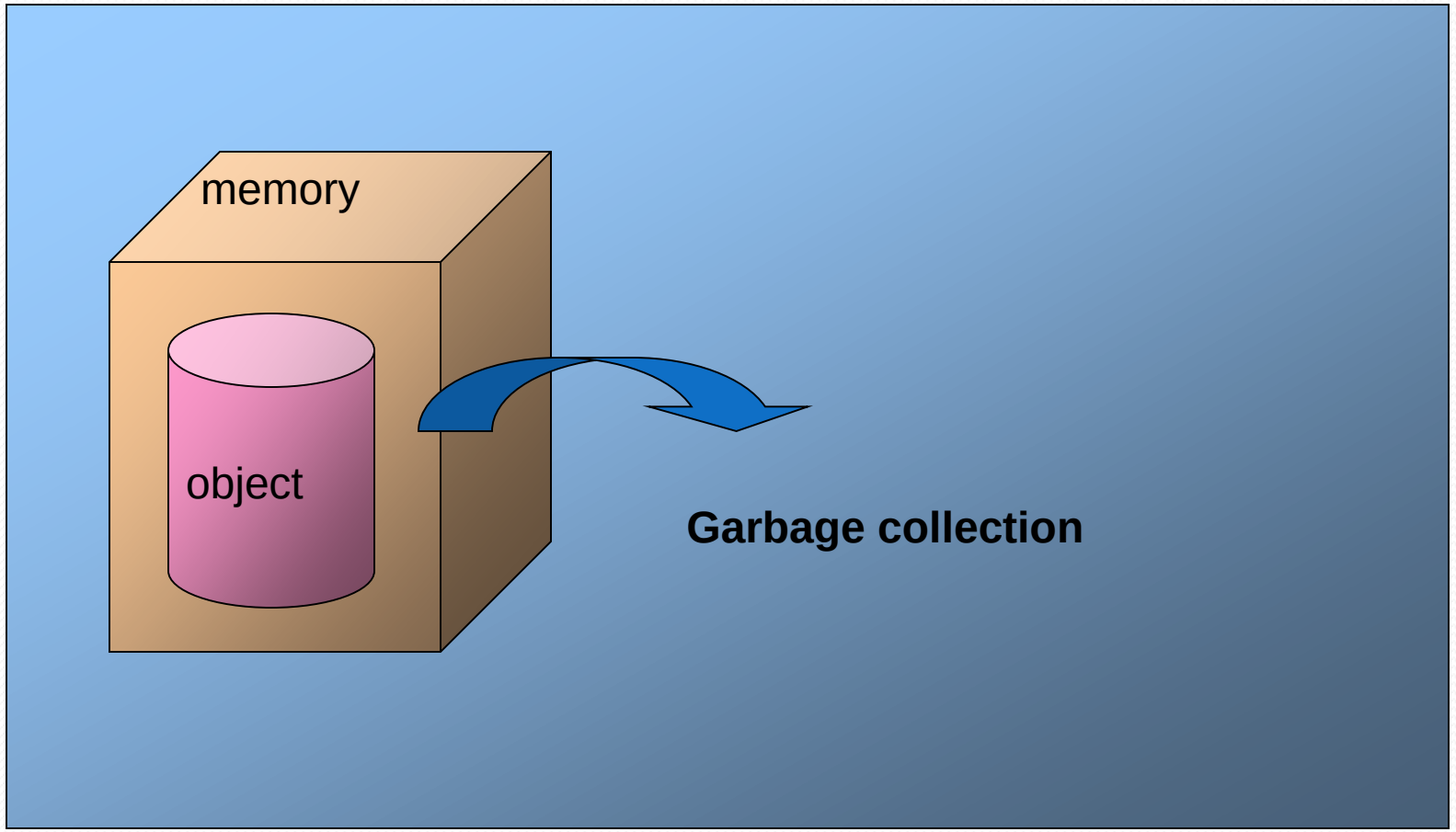
Syntax: **Student stu_ptr=new Student()**

Allocating Memory

```
class Sdate {
 int[] date;
    Sdate(int m,int d,int y) {
      date = new int[3];
       date[0]=m;
      date[1]=d;
      date[2]=y;
    }
 }

class Newdate {
  public static void main(String args[])    {
        Sdate S1,S2;
        S1=new Sdate(11,27,1967);
        S2=new Sdate(4,3,1973);
   }
 }
```

De-allocating Memory



Garbage Collector (1)

- The working of garbage collector is as follows:
  - An object with a destructor defined is added to a list of objects that require destruction.
  - Garbage collector starts on its rounds and checks if there are objects that have no references.
  - If an object is found and if the name of the object does not appear in the finalizer list then it is cleared up instantly.
  - If the name of the object appears on the list of objects that require finalization, it is marked as "Ready for Finalization".

Garbage Collector (2)

- When the garbage collection is complete then the finalizer thread is called, which goes about calling the finalize methods of all objects that have been marked as “Ready for Finalization”.
- After the finalization of an object has occurred, it is removed from the list of objects, which require finalization.
- Since the object is no longer on the finalizer list, it gets cleaned up when the next garbage collection is done.

Garbage Collector (3)

- Objects with finalize method take up more resources as they stay for a longer period in memory even when they are not required.

- Finalization takes place as a separate thread, again eating into the resources.

Finalize method

```
Protected void finalize()
{
//finalization code
}
```

Information Hiding

- Information is stored within the object
- It is hidden from the outside world
- It can only be manipulated by the object itself

Example – Information Hiding

- Ali’s name is stored within his brain
- We can’t access his name directly
- Rather we can ask him to tell his name

Example – Information Hiding

- A phone stores several phone numbers
- We can’t read the numbers directly from the SIM card
- Rather phone-set reads this information for us

Information Hiding Advantages

- Simplifies the model by hiding implementation details
- It is a barrier against change propagation

Encapsulation

- Data and behavior are tightly coupled inside an object
- Both the information structure and implementation details of its operations are hidden from the outer world

Example – Encapsulation

- Ali stores his personal information and knows how to translate it to the desired language

- We don’t know
  - How the data is stored
  - How Ali translates this information

Example – Encapsulation

- A Phone stores phone numbers in digital format and knows how to convert it into human-readable characters

- We don’t know
  - How the data is stored
  - How it is converted to human-readable characters

Encapsulation – Advantages

- Simplicity and clarity
- Low complexity
- Better understanding

Object has an Interface

- An object encapsulates data and behavior
- So how objects interact with each other?
- Each object provides an interface (operations)
- Other objects communicate through this interface

Example – Interface of a Car

- Steer Wheels
- Accelerate
- Change Gear
- Apply Brakes
- Turn Lights On/Off

Example – Interface of a Phone

- Input Number
- Place Call
- Disconnect Call
- Add number to address book
- Remove number
- Update number

Implementation

- Provides services offered by the object interface

- This includes
  - Data structures to hold object state
  - Functionality that provides required services

Example – Implementation of Gear Box

- Data Structure
  - Mechanical structure of gear box
- Functionality
  - Mechanism to change gear

Example – Implementation of Address Book in a Phone

- Data Structure
  - SIM card
- Functionality
  - Read/write circuitry

Separation of Interface & Implementation

- Means change in implementation does not effect object interface
- This is achieved via principles of information hiding and encapsulation

Example – Separation of Interface & Implementation

- A driver can drive a car independent of engine type (petrol, diesel)
- Because interface does not change with the implementation

Example – Separation of Interface & Implementation

- A driver can apply brakes independent of brakes type (simple, disk)
- Again, reason is the same interface

Advantages of Separation

- Users need not to worry about a change until the interface is same
- Low Complexity
- Direct access to information structure of an object can produce errors