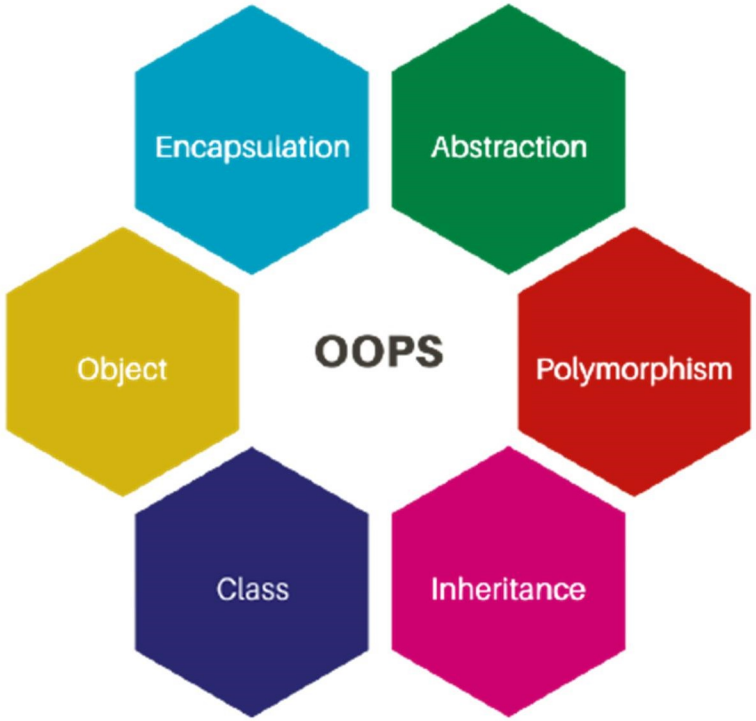


ABESIT
COLLEGE OF ENGINEERING

CRC-Training

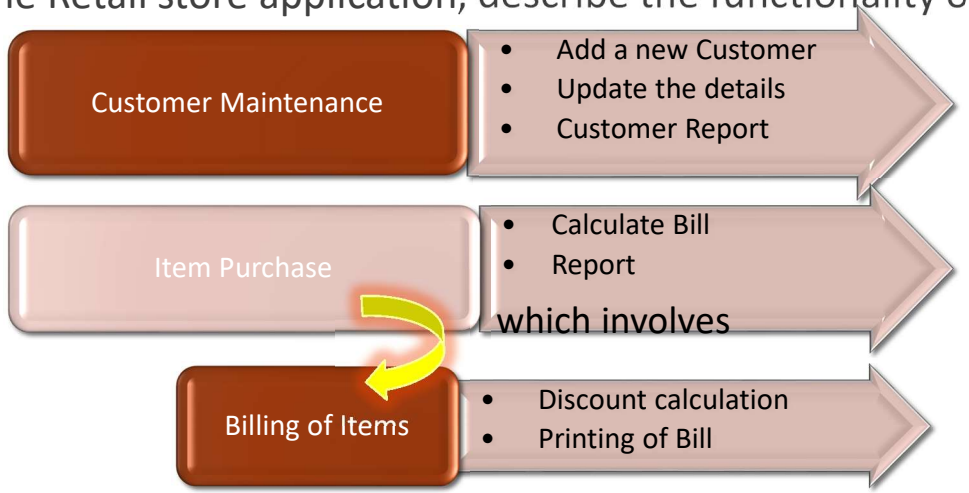
Java™

OOPS Concept

Structured Programming Approach

Retail Application-Activity

Consider the Retail store application, describe the functionality of the system



The focus is on the functionality – This is the structured approach

Advantages-

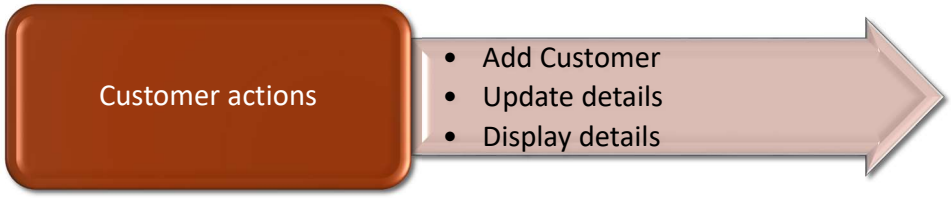
- Each functionality is clearly defined
- The functionality has been sub-divided
- A particular functionality may be changed without affecting the others

Object Oriented Approach

Retail Application-Activity

Consider the following details of a customer being stored -

What are the actions that are possible on this Customer data ?



The focus is on the objects and their behavior – This is the object oriented approach

Advantages-

- Helps create reusable components – ex. The customer component can be reused while there is a purchase of items by customer
- Extends and builds upon the structured programming paradigm

Structured vs. object oriented approach

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Retail Application-Activity

Consider the Customer maintenance functionality of the retail store

Structured Approach Perspective

Customer Maintenance

- Add a new Customer
- Update the details
- Generate Customer Report

Defined in terms of the functionality

Emphasis is on the functionality

May use global data which is not very secure

Object Oriented Approach Perspective

Customer details

- Customer Id
- Customer Name
- Telephone No
- Address

Customer Activities

- Add Customer
- Update details
- Display details

Defined in terms of the attributes and the activities on the customer ie. Customer as an object

Emphasis is on the data and the activities on the data

Has mechanisms to secure the data , termed as “Data Hiding”

Let us look at the Customer object

Object Oriented Approach

Benefits of Object Oriented Approach

- Leads to development of smaller but stable subsystems
- Such subsystems are resilient to change
- Reduces the risk factor in building large systems as they are built incrementally from subsystems which are stable

Hence Object Orientation is suitable for developing extremely complex systems

Object-oriented Programming	Structural Programming
It follows a bottom-up approach.	It follows a top-down approach.
It provides data hiding.	Data hiding is not allowed.
It is used to solve complex problems.	It is used to solve moderate problems.
It allows reusability of code that reduces redundancy of code.	Reusability of code is not allowed.
It is based on objects rather than functions and procedures.	It provides a logical structure to a program in which the program is divided into functions.
It provides more security as it has a data hiding feature.	It provides less security as it does not support the data hiding feature.
More abstraction more flexibility.	Less abstraction less flexibility.
It focuses on data.	It focuses on the process or logical structure.

ABESIT
COLLEGE OF ENGINEERING

CRC-Training

Object Oriented Concepts

Pillars of Object Oriented Programming

Let us now understand :

- Classes and Objects
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Classes & Objects

Retail Application-Activity

Let us look at a customer object

Add customer is an activity on the customer

Values of the attributes

Customer Id:
1001
Customer Name:
John
Telephone Number:
9901911445
Address:
ABESIT , Ghaziabad

Attributes of the Customer

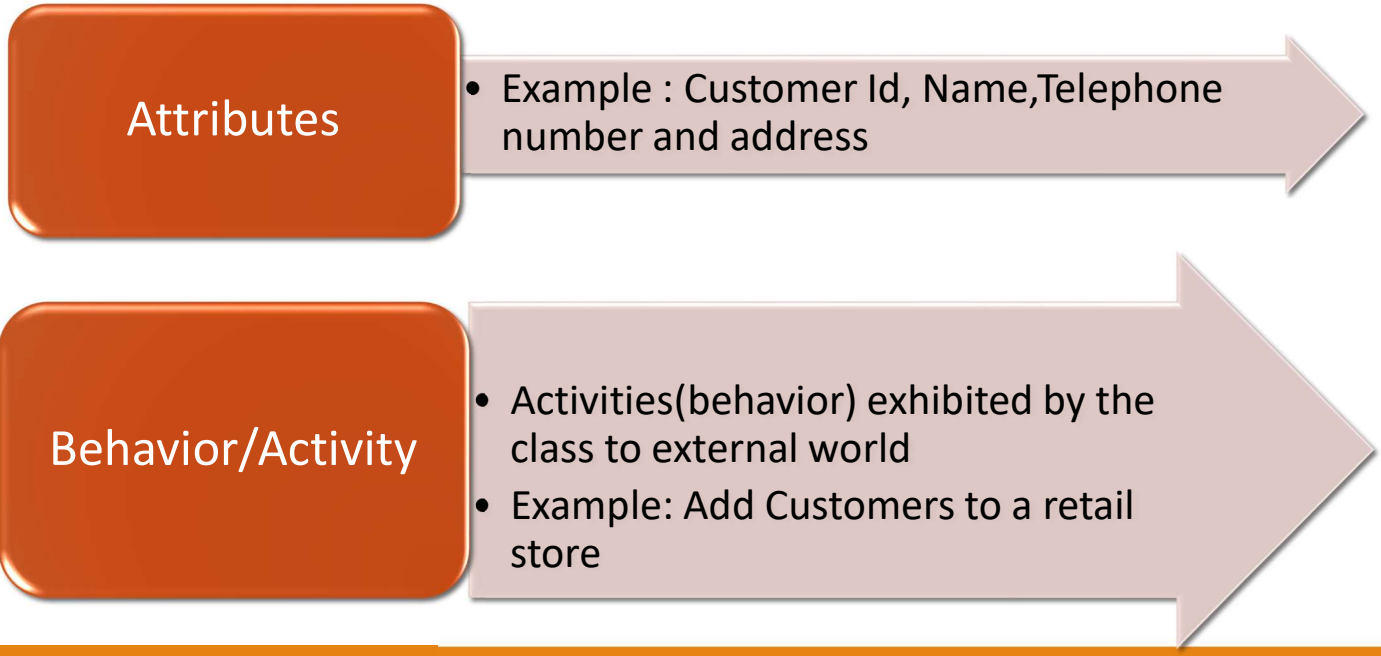
How is the template for Customer objects represented?

Using a Class

Classes & Objects

A class is a software design that describes the common attributes and activities (behavior) of objects

Retail Application-Activity



Classes & Objects

John & Jane are two instances/objects belonging to Customer class

Retail Application-Activity

Attributes of Customer

Customer Id:	**Customer Id:**
1001	1002
Customer Name:	**Customer Name:**
John	Jane
Telephone Number:	**Telephone Number:**
9901911445	9496244655
Address:	**Address:**
Silver Shine, Delhi	Crossing Republic , Meerut

Values of attributes of Customer

ABSTRACTION

Retail Application-Activity

Consider the Retail store application:

When a customer goes to purchase a television for example, what are the details given by the salesperson about the television?

Characteristics of the television (model, make, color etc)
Basic functioning of the television

What are the details which are not mentioned by the salesperson ?

The internal components (resistors, switches etc)
How the internal components are wired and how they work

Let us identify the OO concept discussed here

ABSTRACTION : Process of identifying the essential details to be known and ignoring the non-essential details from the perspective of the end user

Abstraction

Let us look at the users in the retail store application

The bill generator should know the details of the customer and also how to approve and dispatch the bill

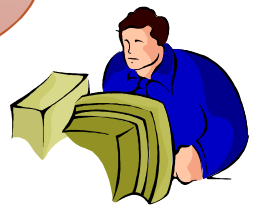
Billing staff

What are the observations here?

Retail Application system

The customer needs to know only what to purchase and the mode of payment

Customers



The administrator knows and has access to the internal working of the system

Administrator

ABSTRACTION : Process of identifying the essential details to be known and ignoring the non-essential details from the perspective of the end user

Abstraction

Defined as the process of focusing the essential details and ignoring the non-essential details

- Helps simplify the understanding and using of any complex system
  - Ex. The driver of a car needs to know how to apply brake, change gear and balance the steering. The driver need not know how the engine works

ENCAPSULATION

Retail Application-Activity

Consider the Retail store application wherein the customer purchases the television.

How does the customer check the working of the television?

The customer uses the switch available in the television box/remote case

The television box /remote case encapsulates the complex working of the machine and provides only a simple interface for operating the television

Let us identify the OO concept discussed here

ENCAPSULATION : A mechanism of hiding the internal details and allowing a simple interface which ensures that the object can be used without having to know how it works

Encapsulation

Let us revisit the payment activity in a retail store using a swipe machine

The billing staff uses the input interface to key in the amount

The customer uses the swipe interface provided by the machine to swipe the card

The swipe machine encapsulates/hides the internal circuitary from all the users and provides simple interfaces for access by every user

The administrator uses the network interface to record the payment

Let us identify the OO concept discussed here

ENCAPSULATION : A mechanism of hiding the internal details and allowing a simple interface which ensures that the object can be used without having to know how it works

Encapsulation

Encapsulate = “En” + “Capsulate”

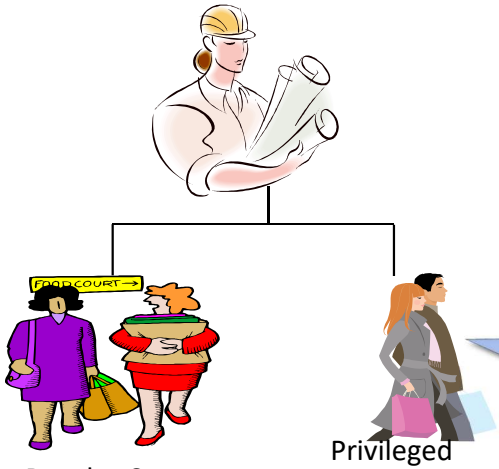
- En = “In a”
- Capsulate = “ Capsule”

- Data hiding

  - Ex. The bonnet of a car encapsulates the car’s engine

Inheritance

Let us revisit the scenario of a customers in the retail store



Regular Customers

Privileged Customers

Let us identify the OO concept discussed here

Customers are of two kinds- Regular customers and Privileged customers

All customers have Customer Id, Name, Telephone Number and Address

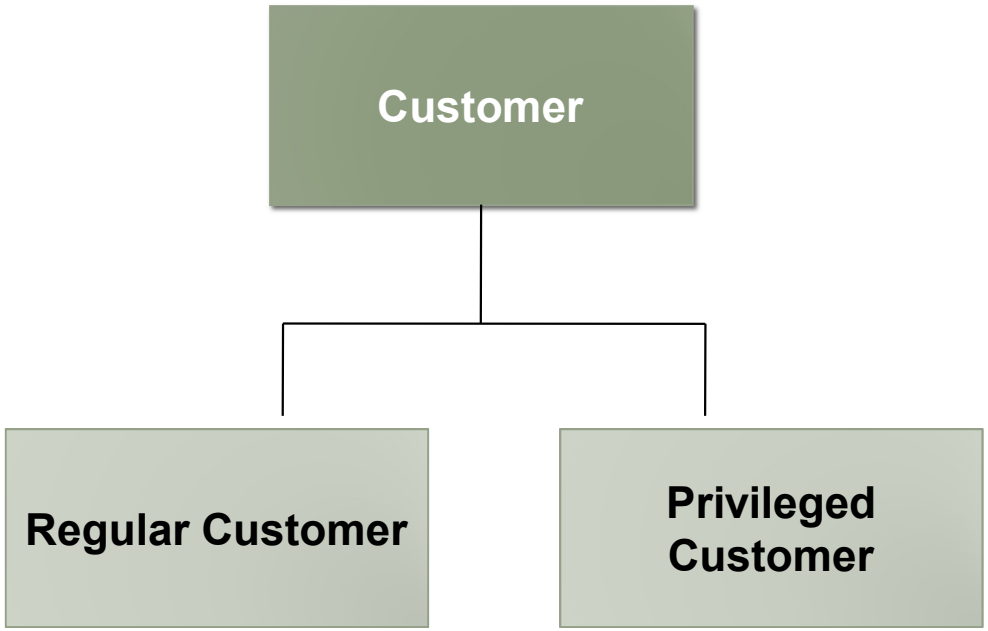
The regular customer in addition is allowed discounts

The privileged customer in addition is allowed to get gifts

INHERITANCE : Is a mechanism which allows to define generalized characteristics and behavior and also create specialized ones. The specialized ones automatically tend to inherit all the properties of the generic ones

Inheritance

Retail Application-Activity



Observations:
Customer is the generalized class
Regular Customer & Privileged Customer are the specialized classes of Customer class

Inheritance

Concept wherein a class shares some common structure or behavior with one or more classes

Classes are arranged in a tree like structure called a hierarchy

Base class:

- The class providing the generalized characteristics and behavior. Ex. Customer class

Derived class:

- The class providing the specialized characteristics and behavior. Ex. Regular Customer & Privileged Customer

Polymorphism

Retail Application-Activity

Consider the Retail store application. A customer has purchased items. The bill is being calculated. Is the calculation of bill same for Regular and Privileged customers?

No!!! Privileged Customers get gifts based on membership card whereas Regular Customers get a discount based on the amount of purchase

Is the calculation of bill needed for all the customers?

Yes, the calculation of bill is needed for for all the customers but the formulae for calculation of bill will differ based on the type of customer i.e the way of calculating the bill differs for different situations

POLYMORPHISM: Refers to the ability of an object/operation to behave differently in different situations

In the above example, if a new category of customers is added, the system would still calculate the bill , but how the bill gets calculated depends on the type of customer added

Let us identify the OO concept discussed here

Polymorphism

Let us visit the scenario of a calculation of bill during purchase

Payment through Credit Card

Total Amount = Purchase amount + VAT + Card Tax

What are the observations here ?

Payment through Cash

Total Amount = Purchase amount + VAT

POLYMORPHISM: Refers to the ability of an object/operation to behave differently in different situations

Let us identify the OO concept discussed here

Polymorphism

Refers to an object's ability to behave differently depending on its type

- Poly = 'many'
- morph = 'form'

Ability to take different forms is called polymorphism

Can you answer these questions?

Q1. In the ATM machine, the customer chooses the operations using a touch screen. The customer need not know the internal working of the ATM machine . Which OO concept(s) can be used in this scenario?

Q2. Consider the following statement: “Vehicles can be of two types viz. Water vehicles and Land vehicles “ . Which OO concept may be used to represent this scenario?

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Relationship between objects

- Aggregation
- Association

Relationship between objects

In real life, objects interact with objects of other classes. Let us look at an example:

Below are some other examples that show how different classes may be related to each other:
•Employee drinks Coffee
•Customer buys a Phone
•College has a Department
•Car has a Wheel

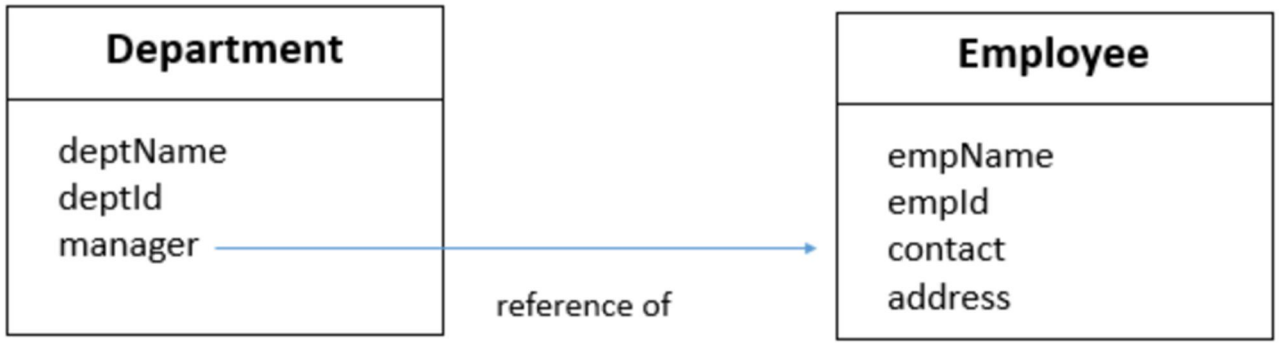
In OOP, two classes can communicate with each other using their objects. An object may communicate with another object to use the functionalities provided by the other object. This is very helpful if we want to reuse the members of a class in another. Some of the types of relationships in Java are:
•Aggregation
•Association
•Inheritance

Aggregation

Aggregation is also known as **Has-a** relationship.

This kind of relationship exists between two classes when a reference variable of one class is a member variable in another class. Consider the below examples.

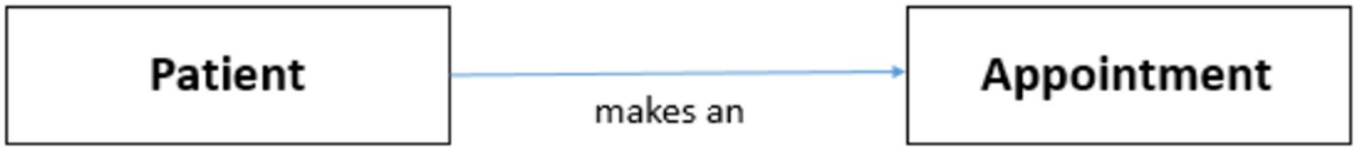
Example 1: - Consider a class Department. Department will have various member variables like department name, department Id. Apart from these member variables, a department also **has a** manager. This manager is actually an Employee and will have different set of member variables. The member variable manager in the Department class will be a reference of Employee class. In this case, we can say, Department **has a** manager (Employee) which is an Aggregation relationship.



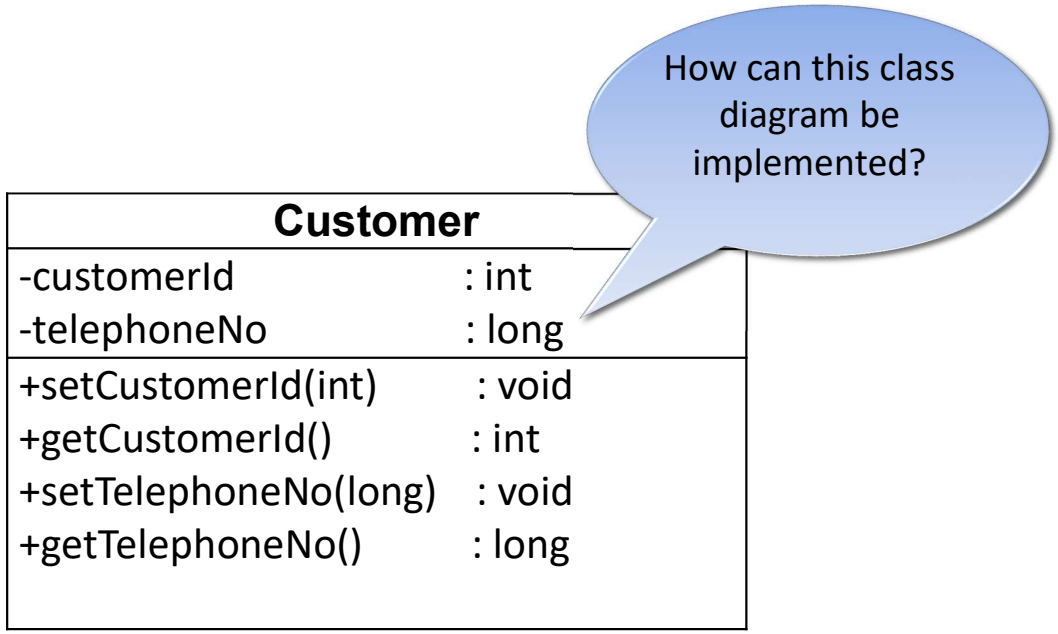
Association

Association, also known as **uses-a** relationship exists between two classes when one object makes use of another object for its functionality. Here, both the objects can exist independently.

Example 1: - Consider a class Patient. A patient needs to take an appointment to visit the hospital. Appointment and Patient are separate classes. Also, an appointment cannot be categorized as one of the attributes of the patient. Rather this appointment is only **used by** the patient and is said to have Association relationship.



Class Diagram



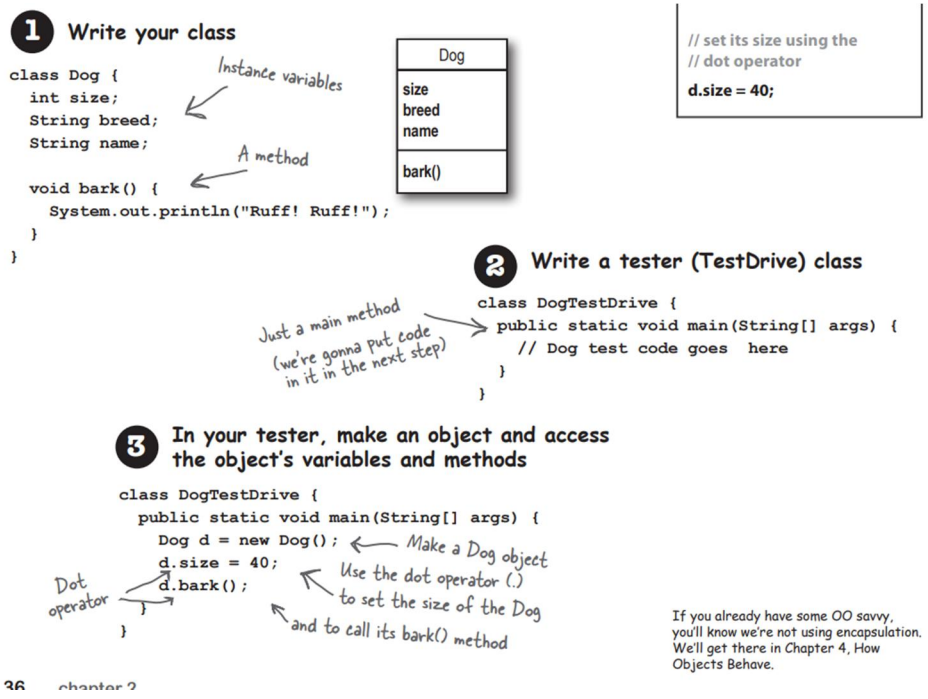
Class Diagram

- First Compartment- Class name
- Second Compartment - State of the class
- Third Compartment - Behavior of the class

Write a class and object creation

ABESIT COLLEGE OF ENGINEERING

CRC-Training



1 Write your class

```
class Dog {
  int size;
  String breed;
  String name;

  void bark() {
    System.out.println("Ruff! Ruff!");
  }
}
```

Instance variables

A method

Dog
size breed name
bark()

```
// set its size using the
// dot operator
d.size = 40;
```

2 Write a tester (TestDrive) class

Just a main method (we're gonna put code in it in the next step)

```
class DogTestDrive {
  public static void main(String[] args) {
    // Dog test code goes  here
  }
}
```

3 In your tester, make an object and access the object's variables and methods

```
class DogTestDrive {
  public static void main(String[] args) {
    Dog d = new Dog();
    d.size = 40;
    d.bark();
  }
}
```

Make a Dog object

Use the dot operator (.) to set the size of the Dog

and to call its bark() method

Dot operator

If you already have some OO savvy, you'll know we're not using encapsulation. We'll get there in Chapter 4, How Objects Behave.

36 chapter 2

Ref: Head First Java

Find the class and no of objects

```
class Movie {
  String title;
  String genre;
  int rating;

  void playIt() {
    System.out.println("Playing the movie");
  }
}

public class MovieTestDrive {
  public static void main(String[] args) {
    Movie one = new Movie();
    one.title = "Gone with the Stock";
    one.genre = "Tragic";
    one.rating = -2;
    Movie two = new Movie();
    two.title = "Lost in Cubicle Space";
    two.genre = "Comedy";
    two.rating = 5;
    two.playIt();
    Movie three = new Movie();
    three.title = "Byte Club";
    three.genre = "Tragic but ultimately uplifting";
    three.rating = 127;
  }
}
```

Ref: Head First Java

Check the code for errors

```
class StreamingSong {

  String title;
  String artist;
  int duration;

  void play() {
    System.out.println("Playing song");
  }

  void printDetails() {
    System.out.println("This is " + title +
                       " by " + artist);
  }
}

class StreamingSongTestDrive {
  public static void main(String[] args) {

    song.artist = "The Beatles";
    song.title = "Come Together";
    song.play();
    song.printDetails();
  }
}
```

Ref: Head First Java

Fill in the blanks

I am compiled from a .java file.	class
My instance variable values can be different from my buddy's values.	object
I behave like a template.	class
I like to do stuff.	object, method
I can have many methods.	class, object
I represent "state."	instance variable
I have behaviors.	object, class
I am located in objects.	method, instance variable
I live on the heap.	object
I am used to create object instances.	class
My state can change.	object, instance variable
I declare methods.	class
I can change at runtime.	object, instance variable

Ref: Head First Java

Check the code for errors

```
class Episode {

  int seriesNumber;
  int episodeNumber;

  void skipIntro() {
    System.out.println("Skipping intro...");
  }

  void skipToNext() {
    System.out.println("Loading next episode...");
  }
}

class EpisodeTestDrive {
  public static void main(String[] args) {

    Episode episode = new Episode();
    episode.seriesNumber = 4;
    episode.play();
    episode.skipIntro();
  }
}
```

Ref: Head First Java

Review Questions

1. Which of the following is not OOPS concept in Java?
a) Inheritance
b) Encapsulation
c) Polymorphism
d) Compilation

2. Which concept of Java is a way of converting real world objects in terms of class?
a) Polymorphism
b) Encapsulation
c) Abstraction
d) Inheritance

3. Which concept of Java is achieved by combining methods and attribute into a class?
a) Encapsulation
b) Inheritance
c) Polymorphism
d) Abstraction

Review Questions

4. Under which pillar of OOPS do base class and derived class relationships come?
a) Inheritance
b) Encapsulation
c) Polymorphism
d) Compilation

5. Why is reusability a desirable feature?
a) Reduces Compilation time
b) Decreases Testing time
c) Lower Maintenance Cost
d) None

6. How do encapsulation and abstraction differ?
a) Hiding and Binding
b) Binding and Hiding
c) Hiding and Hiding
d) None

7. Which among the following feature is not in the general definition of OOPS?
a) Modularity
b) Efficient Code
c) Code reusability
d) Duplicate or Redundant Data

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Interview Questions

Q What is meant by the term OOPs?

OOPs refers to Object-Oriented Programming. It is the programming paradigm that is defined using objects. Objects can be considered as real-world instances of entities like class, that have some characteristics and behaviors.

Q What is the need for OOPs?

There are many reasons why OOPs is mostly preferred, but the most important among them are:

- OOPs helps users to understand the software easily, although they don't know the actual implementation.
- With OOPs, the readability, understandability, and maintainability of the code increase multifold.
- Even very big software can be easily written and managed easily using OOPs.

Q What are some major Object Oriented Programming languages?

The programming languages that use and follow the Object-Oriented Programming paradigm or OOPs, are known as Object-Oriented Programming languages. Some of the major Object-Oriented Programming languages include:

Java , C++ , Javascript ,Python ,PHP ,And many more.

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Interview Questions

Q What are the main features of OOPs?

OOPs or Object Oriented Programming mainly comprises of the below four features, and make sure you don't miss any of these:

Inheritance , Encapsulation , Polymorphism ,Data Abstraction

Q What is a class?

A class can be understood as a template or a blueprint, which contains some values, known as member data or member, and some set of rules, known as behaviors or functions. So when an object is created, it automatically takes the data and functions that are defined in the class. Therefore the class is basically a template or blueprint for objects. Also one can create as many objects as they want based on a class.

For example, first, a car's template is created. Then multiple units of car are created based on that template.

Q. What is an object?

An object refers to the instance of the class, which contains the instance of the members and behaviors defined in the class template. In the real world, an object is an actual entity to which a user interacts, whereas class is just the blueprint for that object. So the objects consume space and have some characteristic behavior.
For example, a specific car.