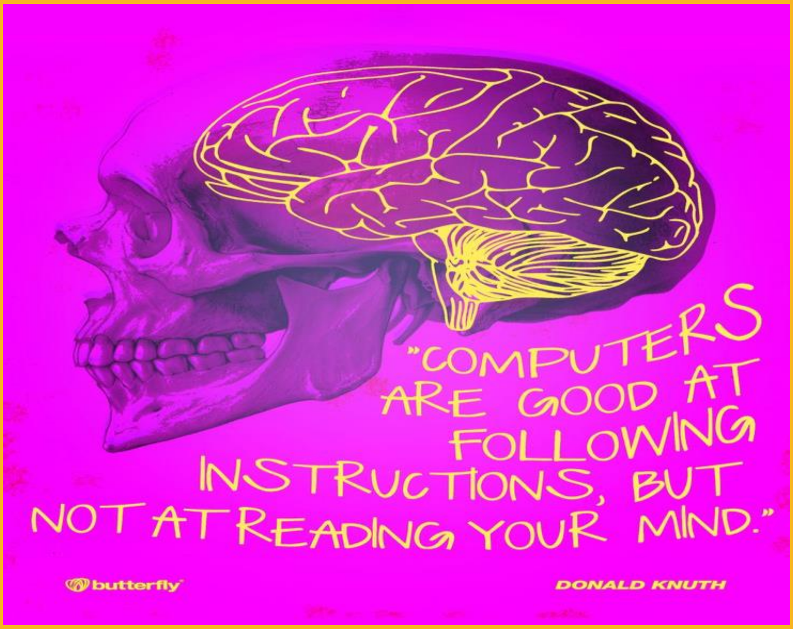

"COMPUTERS ARE GOOD AT FOLLOWING INSTRUCTIONS, BUT NOT AT READING YOUR MIND."
butterfly
DONALD KNUTH

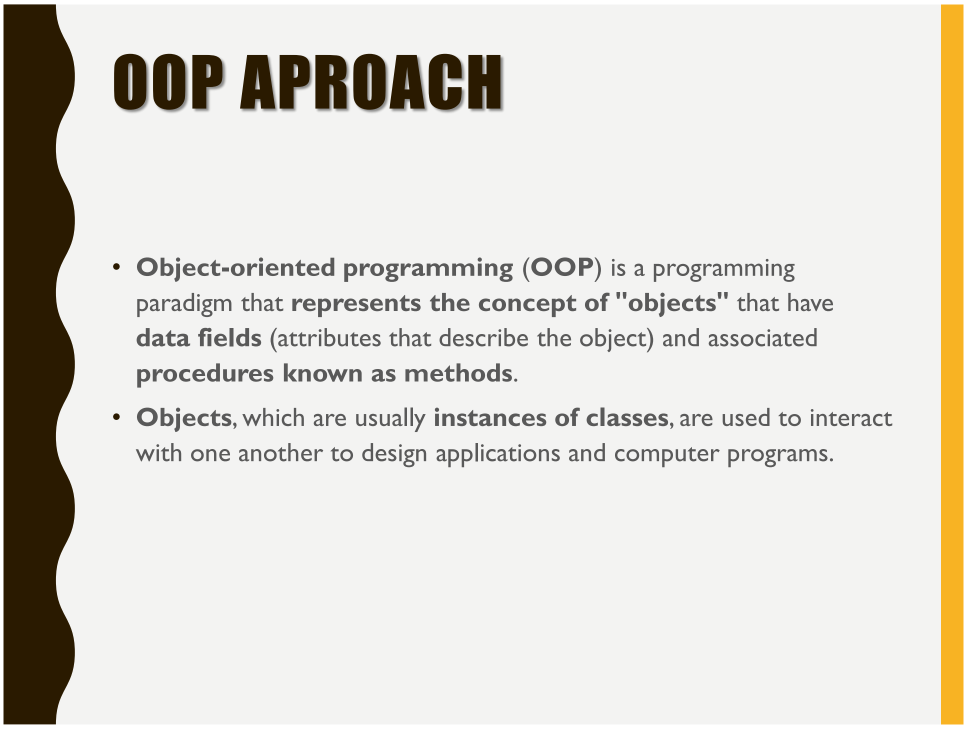
JOB WANTED
?
??

CHAPTER 1:

INTRODUCTION TO
OBJECT ORIENTED PROGRAMMING
(OOP)

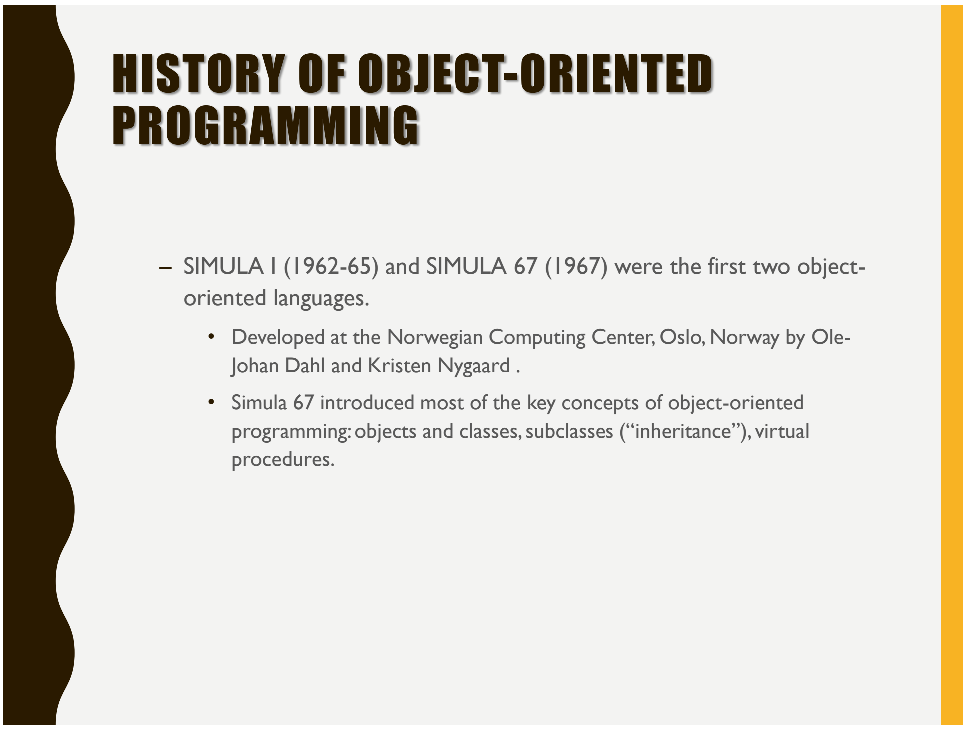
LESSON LEARNING OUTCOME

- Understand the programming techniques
  - Discuss various programming techniques that exist:
    - ❑ Unstructured Programming
    - ❑ Procedural Programming
    - ❑ Object Oriented Programming
  - Describe object oriented approach.
  - Identify the benefits of using OOP approach.
  - Describe the terms used in object oriented analysis and design (OOAD):
    - ❑ Object-Oriented Analysis (OOA)
    - ❑ Object-Oriented Design (OOD)
    - ❑ Object-Oriented Programming (OOP)



OOP APROACH

- **Object-oriented programming** (**OOP**) is a programming paradigm that **represents the concept of "objects"** that have **data fields** (attributes that describe the object) and associated **procedures known as methods**.
- **Objects**, which are usually **instances of classes**, are used to interact with one another to design applications and computer programs.



HISTORY OF OBJECT-ORIENTED PROGRAMMING

- SIMULA I (1962-65) and SIMULA 67 (1967) were the first two object-oriented languages.
  - Developed at the Norwegian Computing Center, Oslo, Norway by Ole-Johan Dahl and Kristen Nygaard .
  - Simula 67 introduced most of the key concepts of object-oriented programming: objects and classes, subclasses ("inheritance"), virtual procedures.

BENEFITS OF USING OBJECT-ORIENTED PROGRAMMING

- Programming for simulation
- Software Crisis
- **Software Reuse**
- Software IC
- Polymorphism,Inheritance and Encapsulation (PIE).
- Classes as an Abstract Data Type(Abstraction)
- **Easy to debug and maintain**
- Mainstream in software development
- Software components.

OBJECT ORIENTED LANGUAGES

- Eiffel (B. Meyer)
- CLOS (D. Bobrow, G. Kiczales)
- SELF (D. Ungar et al.)
- Java (J. Gosling et al.)
- BETA (B. Bruun-Kristensen, O. Lehrmann Madsen, B. Møller-Pedersen, K. Nygaard)
- Other languages add object dialects, such as TurboPascal
- C++ (Bjarne Stroustrup)

ACTIVITY: PLEASE FIND OUT….

OOP

Vs

UNSTRUCTURED PROGRAMMING

VS

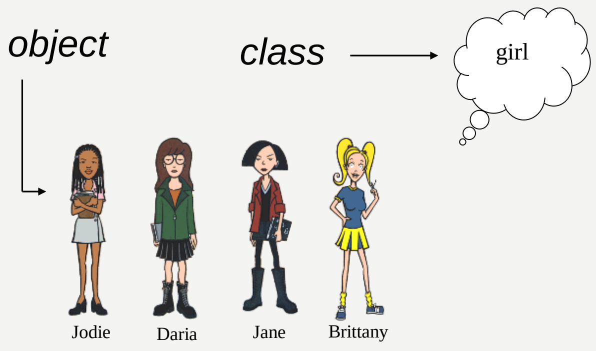
PROCEDURAL PROGRAMMING

LESSON LEARNING OUTCOME

- **Understand the Object Oriented concepts and terminologies**
  - Give example of object oriented languages.
  - Define object, attribute and behaviour.
  - Explain the object oriented terminologies and concepts:
    - Class
    - Object
    - Encapsulation
    - Inheritance
    - Data Abstraction
    - Polymorphism
  - Identify the notation that is used for OOAD by Unified Modelling Language (UML).
  - Describe general UML elements.
  - Apply the Class diagram.
  - Design classes using Class Diagram.

OBJECTS AND CLASSES

- Classes reflect concepts, objects reflect instances that embody those concepts.



- Pn. Hazleena Binti Osman (Jun 2014) -

OBJECT ORIENTED CONCEPTS AND TERMINOLOGIES:

- **CLASS** - A class is a template or blueprint to create an object.
- **OBJECT** - Is an instance of a class.
- Characteristics of real world object are variables or data members in a class - **ATTRIBUTE**.
- **BEHAVIOUR** of objects are called as methods or member functions of a class.

OBJECT ORIENTED CONCEPTS AND TERMINOLOGIES:

- **Data Abstraction** - Refers to the concept of representing only the essential features of a data without including the non-essential details.
- **Encapsulation** - Refers to the mechanism of wrapping up of data and methods (that operate on the data) into a single unit (class).
- **Inheritance** - Refers to the concept by which one class derives the properties of another class.
- **Polymorphism** - Refers to the response (output) of each object differently for the same input.

DIFFERENCE BETWEEN ENCAPSULATION AND ABSTRACTION

- Encapsulate means to hide. Encapsulation is also called data hiding.
- Encapsulation is wrapping, just hiding properties and methods. Encapsulation is used for hide the code and data in a single unit to protect the data from the outside the world.
- Class is the best example of encapsulation.

- Abstraction refers to showing only the necessary details to the intended user. As the name suggests, abstraction is the "abstract form of anything".
- We use abstraction in programming languages to make abstract class. Abstract class represents abstract view of methods and properties of class.

- Pn. Hazleena Binti Osman (December2015)-

OBJECT-ORIENTED ANALYSIS

- Object–Oriented Analysis (OOA) is the **procedure of identifying software engineering requirements** and developing software specifications in terms of a software system's object model, which comprises of interacting objects.
- The main difference between object-oriented analysis and other forms of analysis is that in object-oriented approach, requirements are organized around objects, which integrate both data and functions. They are modelled after real-world objects that the system interacts with. In traditional analysis methodologies, the two aspects - functions and data - are considered separately.

OBJECT-ORIENTED DESIGN

- Object–Oriented Design (OOD) involves **implementation of the conceptual model produced during object-oriented analysis**. In OOD, concepts in the analysis model, which are technology−independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, i.e., a detailed description of how the system is to be built on concrete technologies.

OBJECT-ORIENTED PROGRAMMING

- Object-oriented programming (OOP) is a **programming paradigm** based upon objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

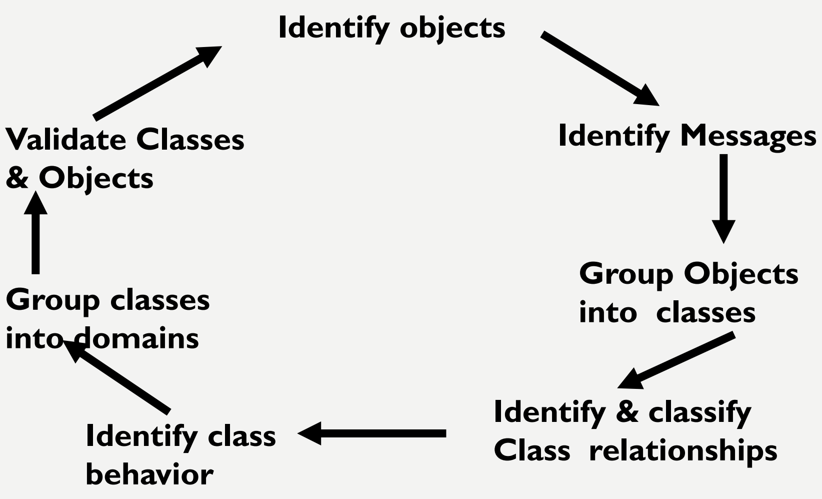
- Pn. Hazleena Binti Osman (December2015)-

OOAD

Object-oriented analysis and design

"Popular **technical approach** to analyse, designing an application, system, or business by applying the object-oriented paradigm and visual modelling throughout the development life cycles to foster better stakeholder communication and product quality."

OOAD---ITERATIVE & INCREMENTAL APPROACH


Identify objects
Identify Messages
Group Objects into classes
Identify & classify Class relationships
Identify class behavior
Group classes into domains
Validate Classes & Objects

WHAT IS UML ?

- UML - Unified Modeling language
- UML is a modeling language, not a methodology or process
- Fuses the concepts of the Booch, OMT, OOSE methods
- Developed by Grady Booch, James Rumbaugh and Ivar Jacobson at Rational Software.
- Accepted as a standard by the Object Management Group (OMG), in 1997.

MORE ON UML…

UML is a modeling language for visualising, specifying, constructing and documenting the artifacts of software systems.

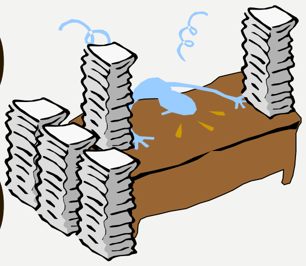
Visualising - *a picture is worth a thousand words; a graphical notation articulates and unambiguously communicates the overall view of the system (problem-domain).*

MORE ON UML…

Specifying - *UML provides the means to model precisely, unambiguously and completely, the system in question.*

Constructing - *models built with UML have a "design" dimension to it; these are language independent and can be implemented in any programming language.*

MORE ON UML…



Documenting - *every software project involves a lot of documentation - from the inception phase to the deliverables.*

Documentation is (among others) for:

- Requirements
- Design
- Tests

UML provides the notations for documenting some of these artifacts

DIAGRAMS

The graphical presentation of the model. Represented as a connected graph - vertices (things) connected by arcs (relationships).

UML includes **nine diagrams** - each capturing a different dimension of a software-system architecture.

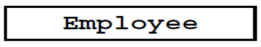
- ◆ Class Diagram
- ◆ *Object Diagram*
- ◆ *Use Case Diagram*
- ◆ *Sequence Diagram*
- ◆ *Collaboration Diagram*
- ◆ *Statechart Diagram*
- ◆ *Activity Diagram*
- ◆ *Component Diagram*
- ◆ *Deployment Diagram*

OBJECTS AND CLASSES CONT'D

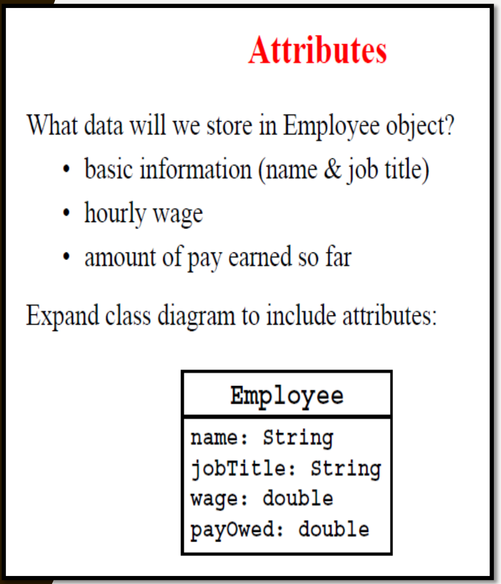
- A **class** captures the common properties of the objects instantiated from it
- A class characterizes the common behavior of all the objects that are its instances

Example Class: Employee

Purpose of class: keep track of an employee for payroll purposes
Very simple class diagram:



No information about attributes & methods



Attributes

What data will we store in Employee object?

- basic information (name & job title)
- hourly wage
- amount of pay earned so far

Expand class diagram to include attributes:

```
Employee
-------------------
name: String
jobTitle: String
wage: double
payOwed: double
```

Java Code :

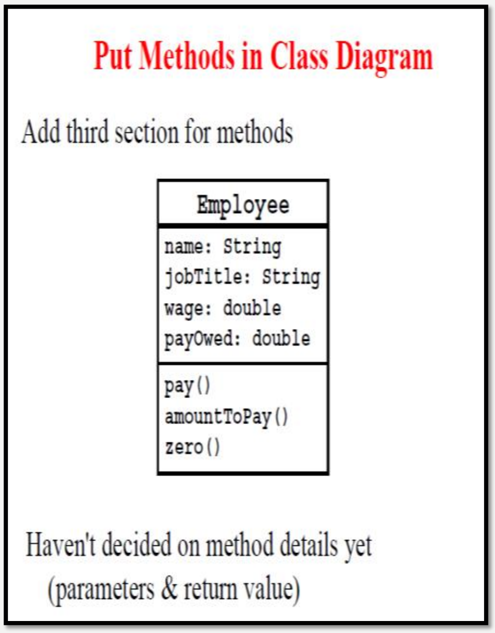
```
/**
* A class to record information about
*employees  for payroll purposes.
*/

public class Employee {
/** the employee's name */
private String name;
/** the employee's job title */
private String jobTitle;
/** the hourly wage for the employee
*/
private double wage;
/** the amount of money the
company currently
owes the employee */
private double payOwed;
}
```

Methods for Employee Class

What do we want to do to/with an Employee?

- create & initialize an Employee object (**constructor**)
- record pay for hours worked
- find out how much we currently owe the employee
- zero the pay owed (after issuingpaycheck)

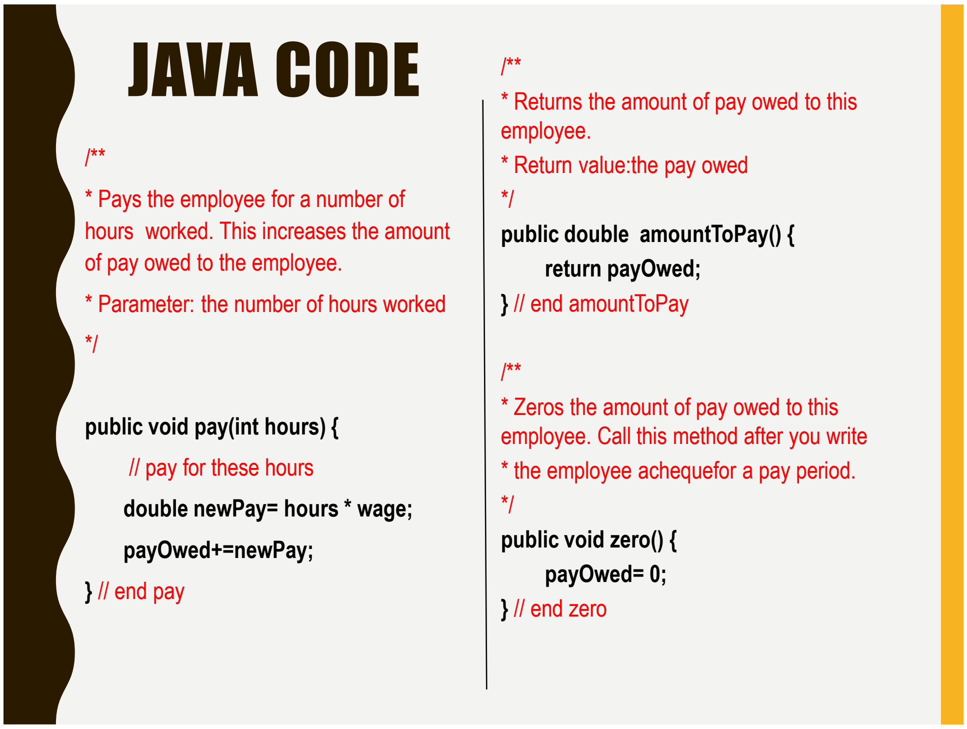


Put Methods in Class Diagram

Add third section for methods

```
Employee
-------------------
name: String
jobTitle: String
wage: double
payOwed: double
-------------------
pay()
amountToPay()
zero()
```

Haven't decided on method details yet
(parameters & return value)



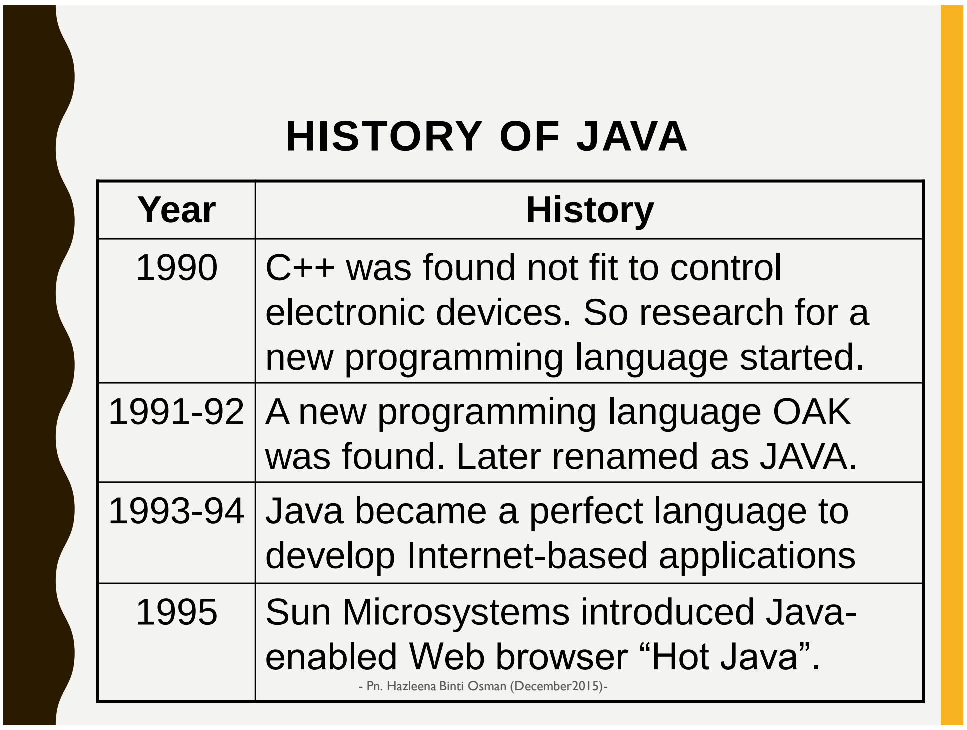
JAVA CODE

```
/**
* Pays the employee for a number of hours  worked. This increases the amount of pay owed to the employee.
* Parameter: the number of hours worked
*/

public void pay(int hours) {
      // pay for these hours
    double newPay= hours * wage;
    payOwed+=newPay;
} // end pay
```

```
/**
* Returns the amount of pay owed to this employee.
* Return value:the pay owed
*/
public double  amountToPay() {
      return payOwed;
} // end amountToPay

/**
* Zeros the amount of pay owed to this employee. Call this method after you write
* the employee achequefor a pay period.
*/
public void zero() {
      payOwed= 0;
} // end zero
```



HISTORY OF JAVA

Year	History
1990	C++ was found not fit to control electronic devices. So research for a new programming language started.
1991-92	A new programming language OAK was found. Later renamed as JAVA.
1993-94	Java became a perfect language to develop Internet-based applications
1995	Sun Microsystems introduced Java-enabled Web browser "Hot Java".

- Pn. Hazleena Binti Osman (December2015)-

JAVA ARCHITECTURE

- Java is platform-independent as it can run on a wide variety of computers using different OS.
- There are four important components in Java architecture:
  - Java Source Code
  - Java Compiler
  - Java Byte code (Object code)
  - Java Virtual Machine (JVM)

JAVA ARCHITECTURE - COMPONENTS

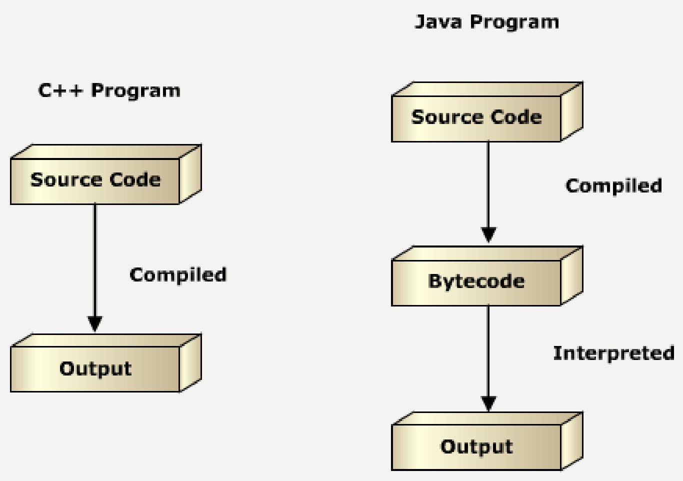
- The components can be explained as follows:
  - **Java Source Code** - Program written in the form of text using Java.
  - **Java Compiler** - Used to convert source code into binary program that consists of byte code. It creates .class file.
  - **Java Byte Code** (Object code) - Byte code is a set of instructions that are machine-independent. Executed by JVM.
  - **Java Virtual Machine** (JVM) - Is a Java runtime system. Converts the bytecode in .class file to machine language.

OBJECT-ORIENTED APPROACH

- Java supports object-oriented approach.
- This approach helps to organize complex programs easily.
- Java implements the real life aspects in programming.

C++ AND JAVA COMPARISON

Compilation model of C++ and Java.


Java Program
C++ Program
Source Code
Source Code
Compiled
Compiled
Bytecode
Interpreted
Output
Output