Project on Android Application

Designed by : **Abhijit Chanda(CSE-32)** & **Mahabuba Rahman Mimma(CSE-60)**
Supervisor: **Kanija Muntarina (Lecturer Dept. of CSE, Dhaka City College)**

Part : 01 **Introduction**
1. Concept of OOP(Object Oriented Programming).
2. Basic Java.
3. Package of Java.

Topics: Concept of OOP

Object oriented programming uses data structures, which consist of methods, data fields, and interactions .It is used to design computer programs and a variety of applications. There are several important techniques, which are used with object oriented programming, including polymorphism, inheritance, modularity, abstraction, and encapsulation. There are a number of new programming languages, which support object-oriented programming.

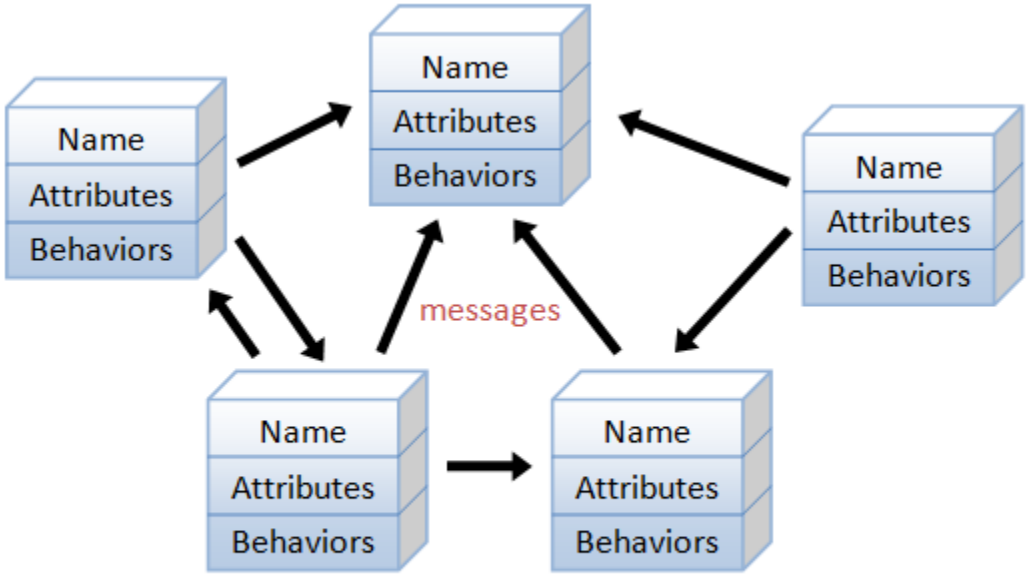
In this type of programming procedures and functions are bundled into a discreet package known as an object . Each of these objects relates to concepts in the real world such as people, vehicles, or accounts. The object is designed in such a way that other software can access it by calling on procedures and functions, which have previously been designated as permissible. This allows for the isolation of these bundled functions and procedures so that they can be more easily managed and tracked.

Object-oriented programming (OOP) languages are designed to overcome these problems.

1. The basic unit of OOP is a *class*, which encapsulates both the *static attributes* and *dynamic behaviors* within a "box", and specifies the public interface for using these boxes. Since the class is well-encapsulated (compared with the function), it is easier to reuse these classes. In other words, OOP combines the data structures and algorithms of a software entity inside the same box.

2. OOP languages permit *higher level of abstraction* for solving real-life problems. The traditional procedural language (such as C and Pascal) forces you to think in terms of

the structure of the computer (e.g. memory bits and bytes, array, decision, loop) rather than thinking in terms of the problem you are trying to solve. The OOP languages (such as Java, C++, C#) let you think in the problem space, and use software objects to represent and abstract entities of the problem space to solve the problem.



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

There are three main features of OOPS.

1) Encapsulation

2) Inheritance

3) Polymorphism

Encapsulation

Encapsulation means putting together all the variables (instance variables) and the methods into a single unit called Class. It also means hiding data and methods within an Object. Encapsulation provides the security that keeps data and methods safe from inadvertent changes. Programmers sometimes refer to encapsulation as using a "black box," or a device that you can use without regard to the internal mechanisms. A programmer can access and use the methods and data contained in the black box but cannot change them. Below example shows Mobile class with properties, which can be set once while creating object using constructor arguments. Properties can be accessed using get() methods which are having public access modifiers.

Inheritance

Inheritance is another important concept in object oriented programming (Harrison et al., 2005).

This refers to the ability to reuse and compartmentalize code by forming a collection of behaviors and attributes for the object. The objects can be placed in classes or subclasses based upon their behavior or attributes. Hierarchies can be created by using a number of nested classes. This concept differs from polymorphism. Inheritance is concerned with the relation among implementations rather than types.

Polymorphism

In Core Java Polymorphism is one of easy concept to understand. Polymorphism definition is that Poly means many and morphos means forms. It describes the feature of languages that allows the same word or symbol to be interpreted correctly in different situations based on the context. There are two types of Polymorphism available in Java. For example, in English the verb “run” means different things if you use it with “a footrace,” a “business,” or “a computer.” You understand the meaning of “run” based on the other words used with it. Object-oriented programs are written so that the methods having same name works differently in different context. Java provides two ways to implement polymorphism.

Benefits of OOP

The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data.

The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data.

JAVA and OOP

In Java, *a class is a definition of objects of the same kind*. In other words, a *class* is a blueprint, template, or prototype that defines and describes the *static attributes* and *dynamic behaviors* common to all objects of the same kind.

An *instance* is *a realization of a particular item of a class*. In other words, an instance is an *instantiation* of a class. All the instances of a class have similar properties, as described in the class definition. For example, you can define a class called "Student" and create three instances of the class "Student" for "Peter", "Paul" and "Pauline".

A Class is a 3-Compartment Box encapsulating Data and Operations

Java Project in OOP!

To understand about Java OOP’S concepts with examples. Let’s discuss about what are the features of Object Oriented Programming. Writing object-oriented programs involves creating classes, creating objects from those classes, and creating applications, which are

stand-alone executable programs that use those objects.

A class is a template, blue print, or contract that defines what an object's data fields and methods will be. An object is an instance of a class. You can create many instances of a class. A Java class uses variables to define data fields and methods to define actions. Additionally, a class provides methods of a special type, known as constructors, which are invoked to create a new object. A constructor can perform any action, but constructors are designed to perform initializing actions, such as initializing the data fields of objects.

Java is inherently *object-oriented,* which means that Java programs are made up of programming elements called *objects.* Simply put, an object is a programming entity that represents either some real-world object or an abstract concept.

All objects have two basic characteristics:

· Objects have data, also known as *state*. For example, an object that represents a book has data such as the book's title, author, and publisher.

· Objects also have *behavior,* which means that they can perform certain tasks. In Java, these tasks are called *methods.* For example, an object that represents a car might have methods such as start, stop, drive, or crash. Some methods simply allow you to access the object's data. For example, a book object might have a get Title method that tells you the book's title.

Why use OOP in java?

Java is a programming language in the tradition of C and C++. As a result, if you have any experience with C or C++, you'll find yourself in familiar territory often as you learn the various features of Java. Java is one of the most popular programming languages used to create Web applications and platforms. It was designed for flexibility, allowing developers to write code that would run on any machine, regardless of architecture or platform. According to the Java home page, more than 1 billion computers and 3 billion mobile phones worldwide run Java.

However, Java differs from other programming languages in a couple of significant ways. The following sections describe the most important differences.

Traditional procedural-oriented languages (such as C and Pascal) suffer some notable drawbacks in creating reusable software components:

1. The programs are made up of functions. Functions are often not *reusable*. It is very difficult to copy a function from one program and reuse in another program because the the function is likely to reference the headers, global variables and other functions. In other words,

functions are not well-encapsulated as a self-contained *reusable unit*.

The procedural languages are not suitable of *high-level abstraction* for solving real life problems. For example, C programs uses constructs such as if-else, for-loop, array, method, pointer, which are low-level and hard to abstract real problems such as a Customer Relationship Management (CRM) system or a computer soccer game. (Imagine using assembly codes, which is a very low level code, to write a computer soccer game. C is better but no much better.

Topics: Basic Java

Java is a simple and yet powerful object oriented programming language and it is in many respects similar to C++. Java was given birth at Sun Microsystems, Inc. in 1991. Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. It was developed to provide a platform-independent programming language.

Java is a generally good OOP language. Sure it has fallen out of grace of the hipster hackers, but it is still one of the most used languages for OOP programming in the corporate world. The late 1980s saw several advances in software development, and by the early 1990s, many large programming projects were being written from prefab components. Java came along in 1995, so it was natural for the language's founders to create a library of reusable code. The library included about 250 programs, including code for dealing with disk files, code for creating windows, and code for passing information over the Internet. Since 1995, this library has grown to include more than 2,700 programs. This library is called the *API* — the *Application Programming Interface.*

What is Object Oriented Programming in Java?

Java is known as an Object Oriented language. So, **what does Object Oriented mean**? It means that the **foundations** of any kind of **program constructed in Java** might be imagined in terms of **Objects**. A good example of this idea should be to have a look at a handful of sample business requirements for a product. Imagine that we are tasked with constructing a software program intended to keep track of an actual **public library system**. This system must keep track of each of the **branches** associated with the libraries, the whole set of **materials** that can be contained in the branches, and additionally all of the **people** that may need to access **books** from the library's branch.

The first thing we're able to do is examine those specs and **pinpoint all of the keywords which happen to be nouns**. For the record, a noun is actually a person, place or thing. Which means, if we analyze our requirements we discern these particular nouns:

1) Library

2) Book

3) Branch

4) Customer

All these words represent Objects in Java. This really is, in essence, Object Oriented programming (generally known as O-O programming). What we can now go about

doing, is in fact arrange these four Objects on to some sort of piece of paper, and begin to distinguish what sort of attributes each one of these Objects contains. What do I mean by the attributes? Well, in O-O development this is known as identifying the “has a” relationships. To provide an example, a Branch “has an” address, a Book “has a” title, a Customer “has a” name. We will **map out the most important important attributes** that each one of these Objects contain, and build ourselves a terrific starting point for the **design** of our Java application

It's OOP principles are solid and you can draw valid comparisons between its design choices and other popular OOP languages like C++ and C# without any difficulty. Moreover, the design patterns applicable in OOP have almost always their Java version.

More importantly, Java to teach OOP because you would come off with the impression that OOP solves every problem and it's a hammer and everything is a nail, when in reality, generic programming and functional programming both have significant advantages in certain situations. Learning about OOP is not just about learning that it *can* be used in X situation or Y situation, but also when it *shouldn't* be applied.

It will also teach that object orientation is linked to other principles- for example, objects can only be heap allocated. What is an object, and where the memory for a type is stored, are two unrelated ideas, but Java couples the two concepts- not to mention how it also couples reference vs value semantics to objects, and other things, which I find ironic, because object orientation is all about decreasing coupling.

To write Java programs, you need three tools:

- **A Java Compiler**
- **A Java Virtual Machine**
- **The Java API**

.

You have at least two ways to get these tools:

· **You can download these tools from the Sun Microsystems Web site.**
· **You can use the tools that come with a commercial product.**

Two bags of goodies
Sun's Web site bundles the basic Java tools in two different ways:

- **The Java Runtime Environment (JRE):** This bundle includes a Java Virtual

Machine and the Application Programming Interface. With the JRE, you can run existing Java programs. That's all. You can't create new Java programs, because you don't have a Java compiler.

- **The Software Development Kit (SDK):** This bundle includes all three tools — a Java compiler, a Java Virtual Machine, and the Application Programming Interface. With the SDK, you can create and run your own Java programs.

Using a customized editor

Even if you don't use an integrated development environment, you can use other tools to make your programming life easy. Think, for a moment, about an ordinary text editor — an editor like Windows Notepad. With Notepad you can

- Create a document that has no formatting
- Find and replace characters, words, and other strings
- Copy, cut, and paste
- Print
- Not much else

How to Use a Constructor in Java

By Doug Lowe from Java For Dummies Quick Reference

A *constructor* in Java is a block of code similar to a method that's called when an instance of an object is created. Here are the key differences between a constructor and a method:

- A constructor doesn't have a return type.
- The name of the constructor must be the same as the name of the class.
- Unlike methods, constructors are not considered members of a class.
- A constructor is called automatically when a new instance of an object is created.

Methods declearation for Java:

The following list describes the method declaration piece by piece:

- visibility: The visibility of a method determines whether the method is available to other classes. The options are

· public: Allows any other class to access the method

· private: Hides the method from other classes

· protected: Lets subclasses use the method but hides the method from other classes

static: This optional keyword declares that the method is a *static method,* which means that you can call it without first creating an instance of the class in which it's defined. The main method must always be static, and any other methods in the class that contains the main method usually should be static as well.

return-type: After the word static comes the *return type,* which indicates whether the method returns a value when it is called — and if so, what type the value is. If the method

doesn’t return a value, specify void.

Sample Code

```
import java.io.*;
public class valuedeclare {
        public static void main(String args[]) throws IOException
        {
        int x,y;
                String s;
                InputStreamReader in=new InputStreamReader(System.in);
                BufferedReader br=new BufferedReader (in);
                System.out.print("Enter the value: ");
                s=br.readLine();
                x=Integer.valueOf(s);
                if(x>0)
                {
                  y=1;
                        System.out.println("The value of y is: "+y);
                }
                else if(x==0)
                {
                  y=0;
                        System.out.println("The value of y is: "+y);
                }
                else
                {
                  y=-1;
                        System.out.println("The value of y is: "+y);
                }
        }
}
```

Topics: Java Package

Packaging and Distributing Java Desktop Applications

Packages are those class which contains methods without the main method in them.Packages are mainly used to reuse the classes which are already created/used in other program.We can define many number of classes in the same package.Packages are mainly divided into two types.They are

1.Built in packages

2.User defined Packages.

The main use of Packages is:

Classes contain in packages of other programme can easily be used.

In packages,classes can be unique,compared with classes in other packages. The package-info.java is a Java file that can be added to any Java source package. Its purpose is to provide a home for package level documentation and package level annotations. Simply create the package-info.java file and add the package declaration that it relates to in the file. In fact, the only thing the package-info.java file must contain is the package declaration**.**

A package is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to different folders on your computer. You might keep HTML pages in one folder, images in another, and scripts or applications in yet another. Because software written in the Java programming language can be composed of hundreds or *thousands* of individual classes, it makes sense to keep things organized by placing related classes and interfaces into packages.

The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications. This library is known as the "Application Programming Interface", or "API" for short. Its packages represent the tasks most commonly associated with general-purpose programming. For example, a String object contains state and behavior for character strings; a File object allows a programmer to easily create, delete, inspect, compare, or modify a file on the filesystem; a Socket object allows for the

creation and use of network sockets; various GUI objects control buttons and checkboxes and anything else related to graphical user interfaces. There are literally thousands of classes to choose from. This allows you, the programmer, to focus on the design of your particular application, rather than the infrastructure required to make it work.

Creating a package:

When creating a package, you should choose a name for the package and put a package statement with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package.

The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

If a package statement is not used then the class, interfaces, enumerations, and annotation types will be put into an unnamed package.

Example:

Let us look at an example that creates a package called **animals**. It is common practice to use lowercased names of packages to avoid any conflicts with the names of classes, interfaces.

Put an interface in the package *animals*:

```
/* File name : Animal.java */
package animals;

interface Animal {
   public void eat();
   public void travel();
}
```

The import Keyword:

If a class wants to use another class in the same package, the package name does not need to be used. Classes in the same package find each other without any special syntax.

Example:

Here, a class named Boss is added to the payroll package that already contains Employee. The Boss can then refer to the Employee class without using the payroll prefix, as demonstrated by the following Boss class.

```
package payroll;

public class Boss
{
  public void payEmployee(Employee e)
  {
    e.mailCheck();
  }
}
```

The Directory Structure of Packages:

Two major results occur when a class is placed in a package:

- The name of the package becomes a part of the name of the class, as we just discussed in the previous section.
- The name of the package must match the directory structure where the corresponding bytecode resides.

For example:

```
// File Name :  Car.java

package vehicle;

public class Car {
  // Class implementation.
}
```

Source Code
Simple 2 java package program which are related one to another

1.secondpack.java

```
package pack02;

public class secondpack {
  int roll;
  String name;
  float mark;

  public void getdata()
  {
     roll=32;
     name="Abhijit";
     mark=752.5f;
  }
  public void putdata()
  {
     System.out.println("Roll is: "+roll);
     System.out.println("Name is: "+name);
     System.out.println("Mark is: "+mark);
  }

}
```

2.**doublepackageprogram.java**

```
import pack02.*;

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

```
        secondpack o=new secondpack();
        o.getdata();
        o.putdata();
    }

}
```