

A Project Report On

**Console based Email Adminstrative Application**

**By:-Raja Kumar**

**Batch:** 2022 – 7400

**Center: Mumbai**

Under the Guidance of,

**Chittaranjan Ghosh.**

**Technical Trainer**

**EduBridge**

(School of coding)

## Introduction:

**1.1 Overview**

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. Java can be used to write applications and applets. A Java application is similar to any other high-level language program: It can only be compiled and then run on the same machine. An applet is compiled on one machine, stored on a server in binary, and can be sent to another machine over the Internet to be interpreted by a Java-aware browser. Java comes with a large library of ready-made classes and objects. The key difference between Java 1.0 and 1.1 was in this library. Similarly, Java 2.0 has a very much larger library for handling user interfaces (Swing by name) but only small changes to the core of the language.

**1.2 Object-oriented Programming**

Java supports object-oriented programming techniques that are based on a hierarchy of classes and well-defined and cooperating objects. Classes: A class is a structure that defines the data and the methods to work on that data. When you write programs in Java, all program data is wrapped in a class, whether it is a class you write or a class you use from the Java API libraries. Classes in the Java API libraries define a set of objects that share a common structure and behavior. Objects: An instance is a synonym for object. A newly created instance has data members and methods as defined by the class for that instance.

# 1.3 Inheritance and Polymorphism:

# One object-oriented concept that helps objects work together is inheritance. Inheritance defines relationships among classes in an object-oriented language. The relationship is one of parent to child where the child or extending class inherits all the attributes (methods and data) of the parent class. In Java, all classes descend from java.lang.Object and inherit its methods. Figure 1 shows the class hierarchy as it descends from java.lang.Objectfor the classes in the user interface example above. The java.Lang. Object methods are also shown because they are inherited and implemented by all of its subclasses, which is every class in the Java API libraries. java.lang. Objectdefines the core set of behaviors that all classes have in common.

# 

**Figure 1. Object Hierarchy**

**1.4 Features of Java**

**Features of Java are as follows:**

1. Compiled and Interpreted

2. Platform Independent and portable

3. Object-oriented

4. Robust and secure

5. Distributed

6. Familiar, simple and small

7. Multithreaded and Interactive

8. High performance

9. Dynamic and Extensible

**BASICS: PACKAGE, CLASS AND OBJECT**

It is important to understand the base terminology of Java in terms of packages, classes and objects. This section gives an overview of these terms.

**Package:**

Java groups classes into functional packages.

Packages are typically used to group classes into logical units. For example all graphical views of an application might be placed in the same package called com vogella. webapplication. views.

It is common practice to use the reverse domain name of the company as top level package. For example the company might own the domain, vogella.com and in this example the Java packages of this company starts with com.vogella.

Other main reason for the usage of packages is to avoid name collisions of classes. A name collision occurs if two programmers give the same fully qualified name to a class. The fully qualified name of a class in Java consists out of the package name followed by a dot (.) and the class name.

Without packages, a programmer may create a Java class called Test. Another programmer may create a class with the same name. With the usage of packages you can tell the system which class to call. For example if the first programmer puts the Test class into package report and the second programmer puts his class into package xmlreader you can distinguish between these classes by using the fully qualified name, e.g. xmlreader. Test or report. Test.

## Topic: - Administrative Email Application

**Objective:-**

**Email Application:-**

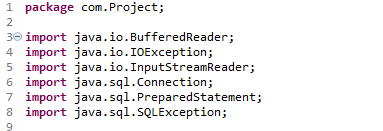
In this email application, learner will be coding the simple email operations like set the mailbox capacity, set the alternate email, change the password, etc. Here learner will be learning about how to generate a random password using Math.random function, how to use OOPS concepts like encapsulation.

**Platform used:-**

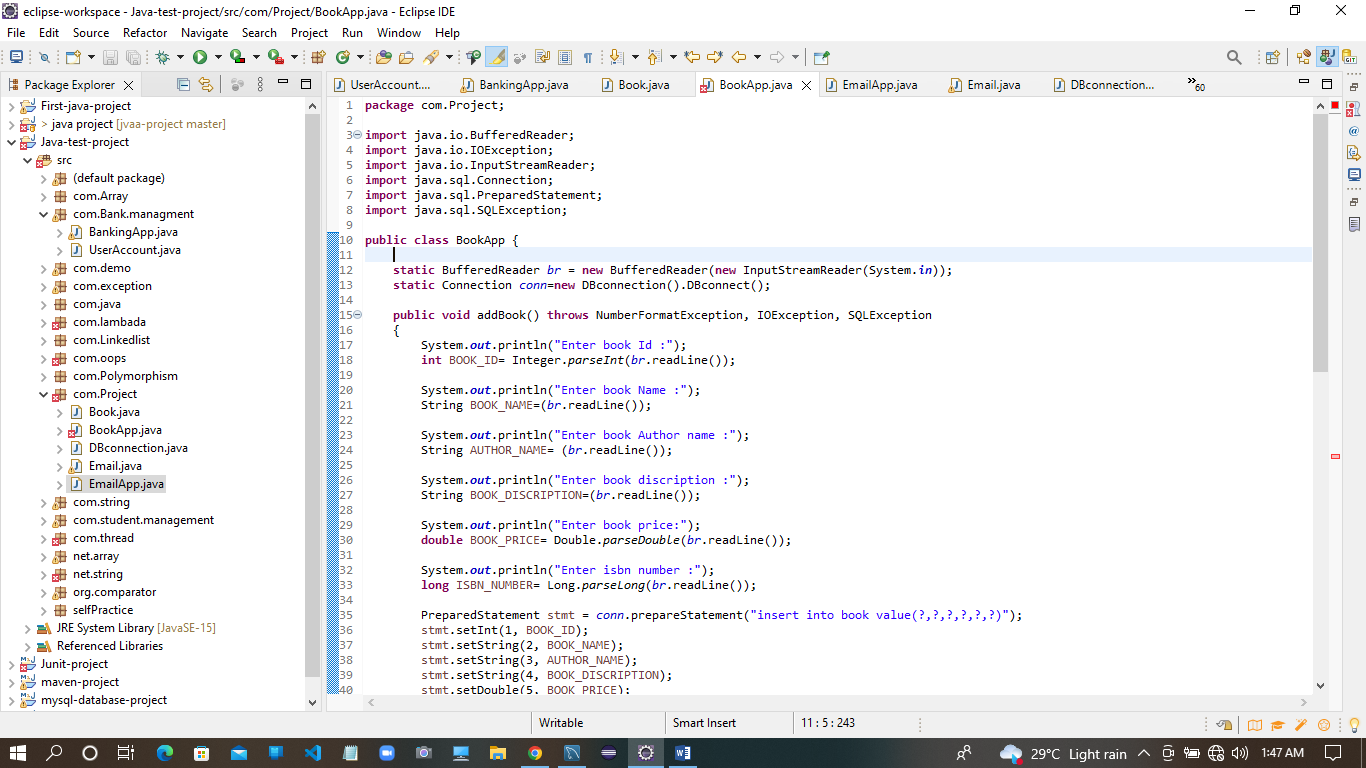
Eclipse IDE



**Screenshots:-**

****

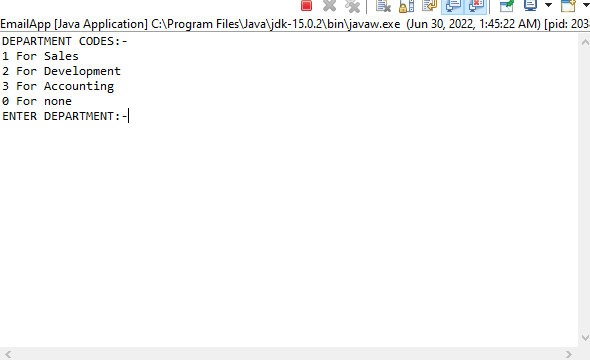
**Figure 2:- Package used in Application**

****

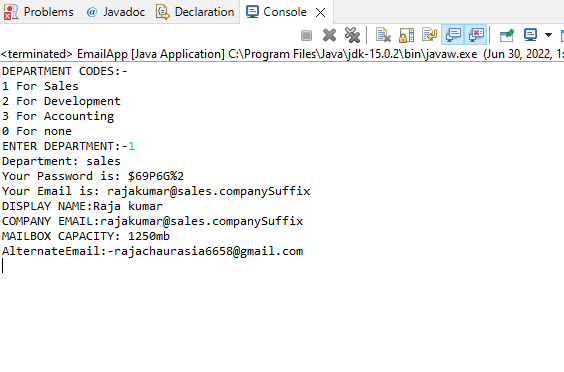
**Figure 3:-source Code**

****

**Figure 4:- Code of Email App**

****

**Figure 5:- Initial O/p for user**

****

**Figure 6:- Final O/p**

**Thank you…..**