**CHAPTER 1**

**INTRODUCTION**

When we talk about a company the first major factor that strikes us or to know about the company is how well the company is performing in terms of revenue that it gets in other terms the economy. An **economy**  is an area of the [production](https://en.wikipedia.org/wiki/Production_(economics)" \o "Production (economics)), [distribution](https://en.wikipedia.org/wiki/Distribution_(economics)" \o "Distribution (economics)) and [trade](https://en.wikipedia.org/wiki/Trade" \o "Trade), as well as [consumption](https://en.wikipedia.org/wiki/Consumption_(economics)" \o "Consumption (economics)) of [goods](https://en.wikipedia.org/wiki/Goods_(economics)" \o "Goods (economics)) and [services](https://en.wikipedia.org/wiki/Service_(economics)" \o "Service (economics)) by different agents in a country produced by the company. The company’s economy can be affected by various factors and the same factors are used to calculate the Economy. GDP is one main major factor in the calculation of Economy. Business Economy of a company is the key term that decides how efficient and productive is it to survive in the competitive world.

**1.1 PROBLEM DEFINITION**

When the company’s economy is in a sluggish state, it is generally harmful for business since consumers and other businesses are less likely to purchase its products. A sluggish economy also has a negative effect on the labour market as business are less willing to hire staff in times of weak economic growth. The factors that are essential for economic growth as well as decline are human resources, physical capital, natural resources and technology. It is difficult to keep in track of all the factors affecting the Economy.

So, this project provides a user-friendly interface in order to keep a track of the economy of a company by using the object-oriented concepts to constantly update the economic situation.

**1.2 OBJECTIVES**

This project mainly concentrates on increasing the efficiency in the way in which the economy of a company is governed. On a larger scale, to keep a track of all the factors affecting the economic status of a company is usually a hectic task. So, this project provides a secured and a simpler way to calculate the budget and keep in track of the economic status of the company without any malfunction or anyone interrupting. It consists of various sectors from which the budget has to be calculated. The entire process flow of calculating and comparing the budget between various companies is explained in the methodology. The final end result of the project helps us study about the compare the economic status of a various companies and they come to a conclusion as to which company stands first in terms of economy for a period of one year.

**1.3 METHODOLOGY TO BE FOLLOWED**

This mini project makes use of the following Object-Oriented Concepts to meet the needs of the following objectives:

* Initially, let’s consider one company whose economy is to be calculated.
* The various factor that contribute to the economic growth of that company, for ex. production, sales, salary etc., is taken as subclasses and few others as super class.
* The values and details required for each factor is taken from the user depending on the parameters required by them.
* In a separate class the total budgeting and revenue is calculated and the economic status of the company is displayed only to the admin.
* The same procedure is followed for several other companies and finally we arrive at a statistical list of all the company’s economic status.

**1.4 EXPECTED OUTCOMES**

There are a variety of end uses of this Economy predictor. This mini project aims to give the following outcomes:

* The main objective is to predict the economic status of a company.
* It helps in providing a well sorted list of the budget details of all the departments in a company which leads to its economy.
* As the project displays the budget and expenses for various department there can never be a threat of scam or indulging in looting money.
* We can know the statistics of each department by department which helps to govern the economy if incase there is a downfall.
* The project leads us to a very useful End result, which is, it displays whether the company is in a profitable economic status or if it is hitting a downfall.
* By this the company can have a trial run of its budget, sales and production and then get to know if that project will be a successful one or not. So it can avoid the risk of downfall.

**1.5 HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware Requirements:**

* **RAM**: 4GB and above
* **Device** **Name**: DELL Vostro 5481 laptop

**Software Requirements:**

* **Operating System**: Microsoft Windows 8 and above
* **Processor**: Processor Intel(R) Core i5-8265 8th Gen
* **Language**: Java
* **Compiler**: Eclipse or NetBeans or IntelliJ IDEA

**CHAPTER 2**

**OBJECT ORIENTED CONCEPTS**

**2.1 CLASS**

A class is basically outline or can also be defined as a user defined blueprint or prototype from which objects are created.  It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

* **Modifiers**: A class can be of four types in terms of visibility mode i.e., public, protected, private and default which are responsible for the security or visibility of the function.
* **Class name:** The name should begin with an initial letter which will consist of various methods, variables and class members.
* **Superclass:** It is basically the parent class, if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
* **Interfaces:** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
* **Body:** The class body surrounded by braces { } which is the main part of the program consisting of code and logical functions.

**2.2 OBJECT**

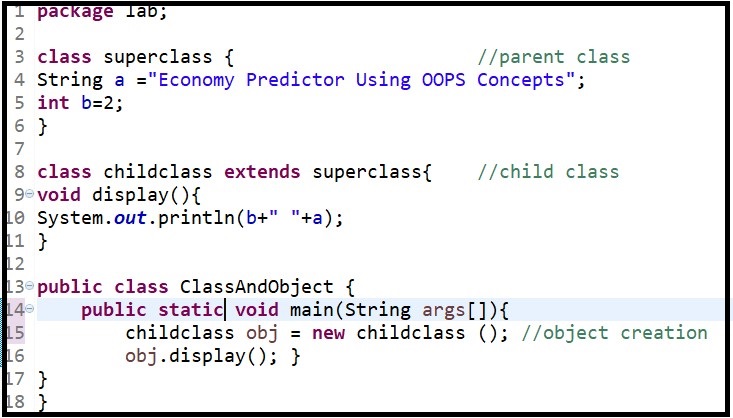
An object is an instance of a class. It is a basic unit of Object-Oriented Programming and represents the real-life entities.  A typical Java program creates many objects, which interact by invoking methods. An object consists of:

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behaviour**: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

An object to a class is created using the keyword “new”. When an object of a class is created, the class is said to be **instantiated**. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

Syntax for creating an object:

Class\_name object\_name = new class\_name ();

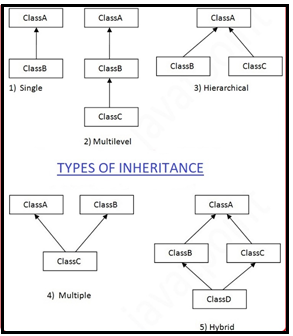


**Fig 2.2.1 Creation of Class and object**

**2.3 INHERITANCE**

Inheritance is one of the major concepts in OOP (Object Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features (methods) of another class thus making it one of the important pillars of OOPS. This leads to the reusability.  
Terminologies involved in inheritance are:

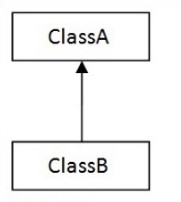
* **Super** **Class**:The class whose features are inherited is known as super class (or a base class or a parent class).
* **Sub Class:** The class that inherits the other class is known as sub class (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
* **Reusability:**Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of inherited class.



**Fig 2.3 Types of inheritance**

**2.3.1 SINGLE INHERITANCE:**

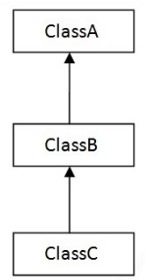
Single inheritance is a feature of object-oriented concept, which refers to a child and parent class relationship where a class extends another class i.e. child class extends its parent class.



**Fig 2.3.1 Single Inheritance**

**2.3.2 MULTILEVEL INHERITANCE:**

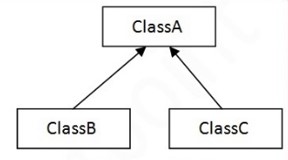
Multilevel inheritance is defined as, if a class is derived from a class which is also derived from another class, i.e. a class having more than one parent classes, then it follows multilevel inheritance. The level of inheritance can be extended to any number of levels depending upon the relation.



**Fig 2.3.2 Multilevel Inheritance**

**2.3.3 HIERARCHICAL INHERITANCE:**

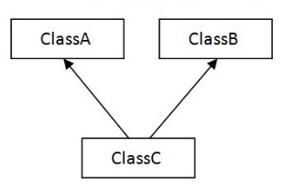
Hierarchical inheritance is a feature of object-oriented concept, in which two or more classes inherits a single class. The class from which inherits the attributes and behaviors are called parent and the class which inherits the attributes and behaviors are called child. In this type the multiple child classes inherit the single class or the single class is inherited by multiple child class.



**Fig 2.3.3 Hierarchical Inheritance**

**2.3.4 MUTLIPLE INHERITANCE:**

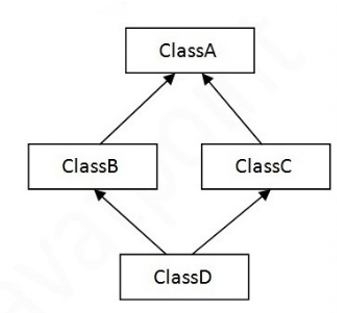
 Multiple Inheritance is a feature of object-oriented concept, which refers to a child and parent class relationship where a class can inherit properties of more than one parent class. This type of inheritance is not supported in java and hence we use interfaces as its substitute.

****

**Fig 2.3.4 Multiple Inheritance**

**2.3.5 HYBRID INHERITANCE:**

Hybrid inheritance is a combination of both single as well as multiple inheritance. Since multiple inheritance is not supported in java, even this type of inheritance is not possible in java.



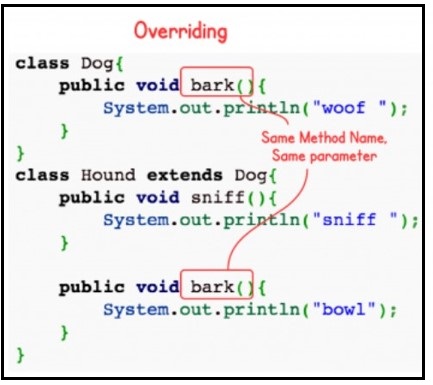
**Fig 2.3.5 Hybrid Inheritance**

**2.4 POLYMORPHISM**

The word polymorphism means having many forms. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, so it means many forms.

There are concepts which explains polymorphism in better way:

* **Method Overloading**: When there are multiple methods with same name but different parameters then these functions are said to be **overloaded**. Functions can be overloaded by **change in number of arguments**or/and**change in type of arguments.**



**Fig 2.4.1 Method overriding**

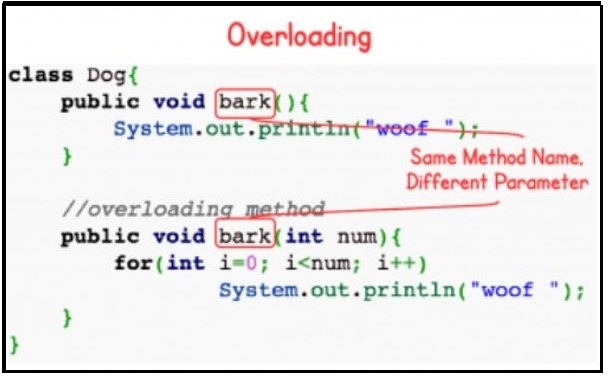
* **Operator Overloading**: Java also provide option to overload operators. For example, we can make the operator (‘+’) for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So, a single operator ‘+’ when placed between integer operands, adds them and when placed between string operands, concatenates them.

In java, only “+” operator can be overloaded:

\*To add integers

\*To concatenate strings

* **Method Overriding**: In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

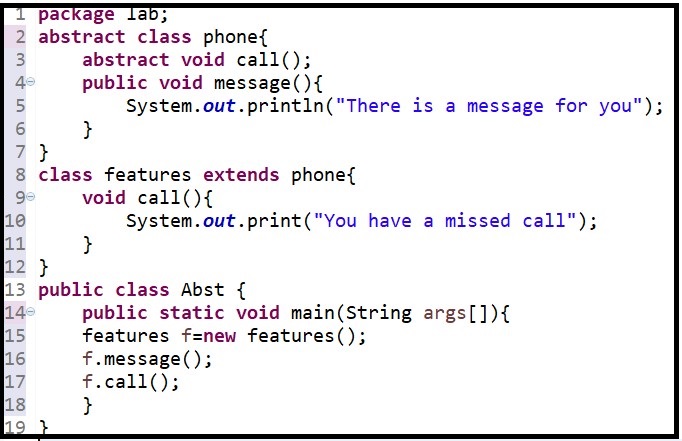
****

**Fig 2.4.2 Method overloading**

**2.5 ABSTRACT CLASS**

Abstract class is a class which is declared with a keyword called as abstract. It is similar to interface except that it can contain default method implementation. Abstract class can have an abstract method without body and it can have methods with implementation also. Abstract classes need to be extended and its abstract method has to be implemented in the extended class.

Abstract class cannot be instantiated i.e. we cannot create an object for abstract class but instead we have to create an object for the class which extends the abstract class and then access its methods. It can have final methods which will force the subclass not to change the body of the method. The subclass of abstract class in java must implement all the abstract methods unless the subclass is also an abstract class.



**Fig 2.5.1 Abstract class**

**2.6 MULTITHREADING**

A thread, in the context of Java, is the path followed when executing a program. All Java programs have at least one thread, known as the main thread, which is created by the Java Virtual Machine (JVM) at the program’s start, when the main method is invoked with the main thread.

Multithreading is one of the features in java that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

There are various methods supported in java for the proper functioning of Thread:

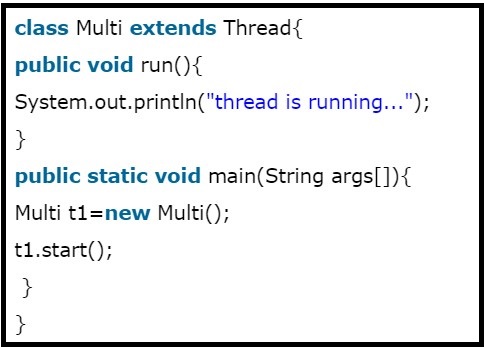
* **setName():** This method is used to set a name for the Thread.
* **getName():** This method is used to return the name of the Thread.
* **getPriority():** It is used to return the priority of the current running Thread.
* **isAlive():** This function is used to check if the Thread is still alive or not.
* run(): This method provides an action to be performed by the Thread.
* sleep(milliseconds): This method causes the currently running thread to block for at least the specified number of milliseconds.

Java defines how the Thread can be created in two ways:

* **By extending the Thread class:**

By this method we can create a thread, in which a class has to extends the Thread class and then create an instance of that class. Now with the object we have to call the start() which implicitly call the run(). The run() which is defined in this class is overriding the run() in Thread class.

* **java.lang.Thread** is used to create thread using thread class

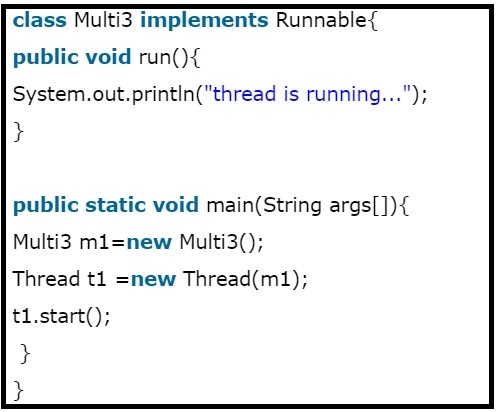


**Fig 2.6.1 Creation of thread by extending Thread class**

* **By implementing the Runnable interface**:

This method is also used to create a thread, in which a class has to implement the Runnable interface. Then an instance of that class is created. This instance of the class cannot be used to directly run the thread but instead another instance for the Thread class is created and the object of the class is passed as reference to the thread class object.

* **java.lang.Runnable** is an interface that is to be implemented by a class whose instances are intended to be executed by a thread.



**Fig 2.6.2 Creation of thread by Implementing Runnable Interface**

* 1. **I/O FUNCTIONS**

Java I/O (Input and Output) is used to process the input from the user and produce the output. Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

**Streams**- A stream is an abstraction that either produces or accepts information. A stream is linked to a physical device by the Java I/O system.

**Byte Streams** - Byte streams provide a convenient means for handling input and output of bytes. Byte streams are used, for example, when reading or writing binary data.

#### Some important Byte stream classes:

* **BufferedInputStream**: This class is used for buffered input stream.
* **DataInputStream**: It contains method for reading java standard datatype.
* **FileInputStream**: It is used to read data from a specified file.
* **InputStream**: Abstract class that describes stream input.
* **BufferedOutputStream**: This class is used for buffered output stream.
* **DataOutputStream**: It is an output stream that contain method for writing java standard data type.
* **FileOutputStream**: It is used to write contents into a file
* **OutputStream**: Abstract class that describes stream output.

**Character Streams** - Character streams provide a convenient means for handling input and output of characters.

#### Some important Byte stream classes:

* **BufferedReader:** It is used to handle buffered input streams.
* **FileReader:** Input stream that reads from file.
* **InputStreamReader:** Input stream that translates byte to character.
* **PrintWriter:** Output stream that contains print() and println() methods.
* **BufferedWriter:** Handles buffered output streams.
* **FileWriter:** Output stream that writes into a file.
* **OutputStreamReader:** Output stream that translates byte to character.

In Java, 3 streams are created automatically. All these streams are attached with the console.

* System.out: standard output stream
* System.in: standard input stream
* System.err: standard error stream

**Reading Console Input**

* In Java the Console input is done by reading from the **System.in**. To obtain the character stream that is entered by the user through the console is read with the help of System.in in the BufferedReader object.

**BufferedReader(Reader inputReader)**

* The InputReader is a stream that is linked to an instance of BufferedReader that is being created. One of the subclasses is InputStreamReader, which will convert the bytes to characters.

**InputStreamReader(InputStream inputStream)**

* To read the character from a BufferedReader, we will use **read( )** function.
* To read the string from console, **readLine( )** function is used, which is the member of a BufferedReader class.

**Writing Console Output**

* PrintStream is the output stream derived from OutputStream. It will also implement the low level method **write( ).** Thus, write( ) can be used to write the console. The simple form of write( )is shown below

**void write(int byte)**

* PrintWriter class will support the **print( )** and **println( )** method. Thus, we can use these methods in the same way that we have used them with System**.**out. If an argument is not simple then the PrintWriter methods will call the object’s to String**( )** method and then will display the result .

**Reading from and writing into a file:**

There are two stream classes which are **FileInputStream** and **FileOutputStream**, which will create byte streams linked to files. To open the file, we simply can create an object **of** one of these classes, specify the name of the file as a argument to the constructor.

* FileInputStream(String fileName)
* FileOutputStream(String fileName)

**2.8 JAVA PACKAGES**

Packages are containers for classes that are used to keep the class name space compartmentalized. In other terms It is a mechanism for partitioning the class name space into more manageable chunks. The package is both a naming and a visibility control mechanism.

**Defining a Package**

To create the package, we simply include the package command as a first statement in a Java source file.

* Any classes that is declared within the file will belong to a specified package.
* If we will skip the package statement then the class names are into the default package that has no name.

The general form of the package:

Package myfirstpackage;

myfirstpackage is the name of the package which will include all the set of classes and methods.

**Types of packages:**

* **Built-in packages**
* **User-defined packages**

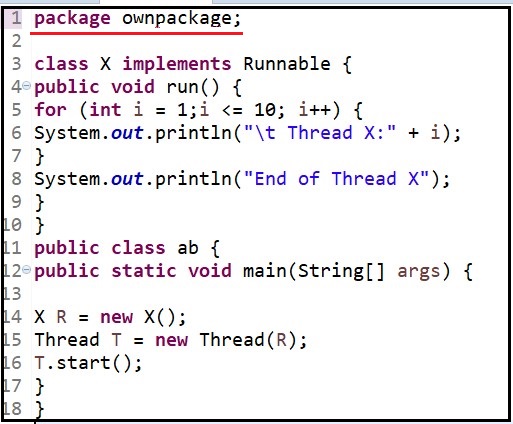
**Built-in packages:**

These packages consist of the large number of classes which have pre-defined classes and functions. Some of the commonly used built-in packages are:

* **java.lang:**Contains language supported classes. This package will be automatically imported.
* **java.io:** Contains classes for supporting input / output operations.
* **java.util:**Contains the utility classes which will implement the data structures like Linked List, Dictionary and support and for the Date / Time operations.
* **java.awt:**Contain the classes for implementing the components for GUI.

**User-defined packages:**

**If the user wants a separate set of packages with different set of their own definition of classes, then this type of package is called User-defined packages.**

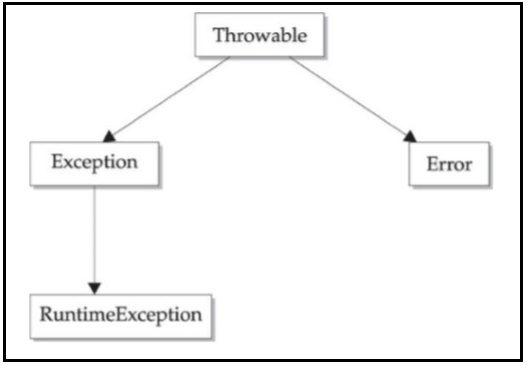


**Fig 2.8.1 User-defined package**

In the above figure 2.8.1 ownpackage is the user-defined package which consist of set of classes and methods and can be imported by other packages by the import keyword.

**2.9 EXCEPTION HANDLING**

A Java exception is an object that defines an exceptional condition that has occurred in a piece of code. When we encounter an exceptional condition, an object representing that exception is created and thrown in the method that caused the error. That method may choose to handle the exception itself, or pass it on. Either way, the exception is caught and processed. Exceptions can be generated by the Java run-time system, or they can be manually generated by code.

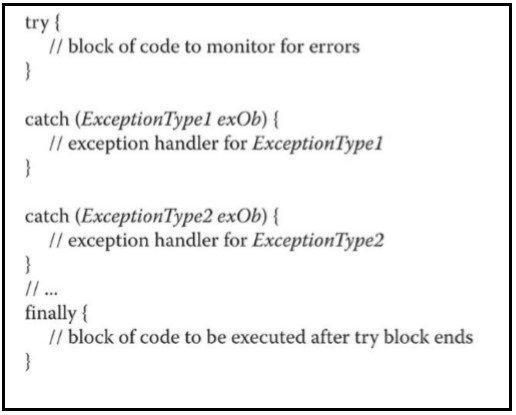


**Fig 2.9.1 Top-level hierarchy of Exceptions**

Exceptions are of two type:

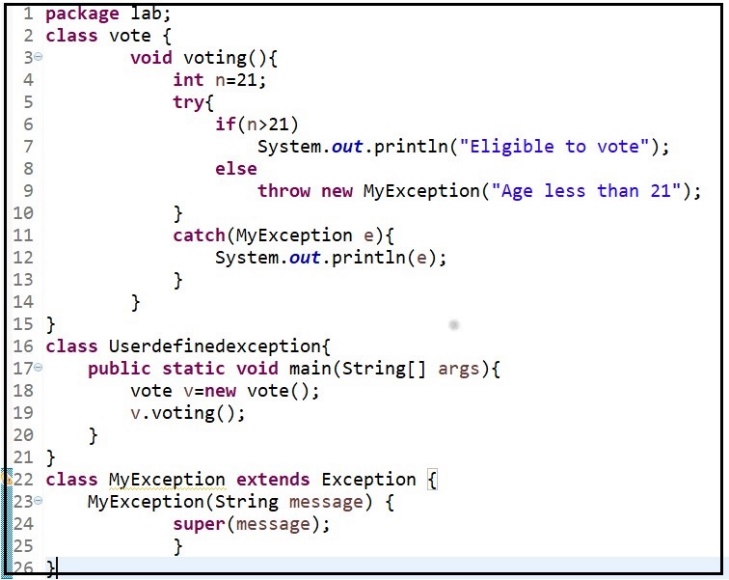
* **Built-in Exception**: Inside the standard package java.lang, Java defines several exception classes. These are exceptions which are suitable to explain error. In the language of Java, these are called unchecked exceptions because the compiler does not check to see if a method handles or throws these exceptions. The following are the list of few important built-in exceptions in Java:
* **ArithematicException**: This type of exception arises when we try divide a number by zero.
* **ArrayIndexOutOfBoundException**: When the array index exceeds the specified limit, then this type of exception is thrown by the method.
* **ClassCastException**: When the casting of object of variable is done incorrectly then this type of exception is thrown by the method.
* **NegativeArraySizeException**: This exception occurs when the array size is given in negative.
* **NullPointerException**: Invalid use of null reference leads to this type of exception.

Java exception handling is managed via five keywords: try, catch, throw, throws, and finally.



**Fig 2.9.2 Exception handling (Built-in Exception)**

* **User-defined Exception:**

****

**Fig 2.9.3 Exception handling (User-defined Exception)**

The figure 2.9.3 shows an example of user-defined exception handling. In rare cases, the built-in exceptions in Java will not be able to describe certain situation. In such cases, the user can also create exceptions which is called User-defined Exceptions.  
User can create their own Exception class which extends Exception class. They can define any Exception handling action under this class which will be called by any class if in case any such Exception occurs similar to the one defined in the user define class.

**CHAPTER 3**

**DESIGN**

**1.DESIGN GOAL:**

In the process of creating this application we are using various concepts provide and supported by JAVA. Some of that which we commonly used are classes, object, inheritance, polymorphism, Java packages, I/O interfaces, multithreading concepts and exception handling. The project deals with the economy of a company. A company consist of administration, customer service, sales and production department. Each of the department are taken as a class and their respective class consist of method to calculate salary. Production class consist of a method which deals with the products to be manufactured and its price and quantity to be produced. Sales class consist of method to handle the exports of the product and simultaneously keeps a track of the revenue earned by exports. Finally, a feedback is taken from the user regarding the quality of the product.

**2. ALGORITHM/PSEUDO CODE:**

* 1. **Calculating the Administrative and customer service departments salaries:**

Step 1: START

Step 2: Create a class **customerservice** which consist of methods **admin()** and **custom()**.

Step 3: Take the input “n” and “m” from the user in terms of number of employees in both

the departments.

Step 4: With few static variables which are initialised with the salary of one employee we

call the method **salarycalculator()** in the **Salarycalculate** class and then finally calculate the salary for n and m employees respectively.

Step 5: Create an abstract class administration which is used to help in overriding the the

admin () in the customerservice class. Overriding is included if in case we have to

increase the salary after a year for the administration department.

Step 6: END

* 1. **Production of products and calculating the production department salary**

Step 1: Start

Step 2: Create a class **produce** which is responsible for calculating budget of production

and salary.

Step 3: Create a method **products()** in which the user will enter the two product names,

raw material cost, price of one product and the number of days within which it

has to be manufactured. According to the above criteria calculate the total

products that can be produced and the total cost.

Step 4: Create a method **salaryproduction()** the salary is calculated for n number of

employees given by the user.

Step 5: END

* 1. **Calculating the salary for Sales department and managing the exports and sales of products:**

Step 1: START

Step 2: Create a class **sales** which extends **produce** class. The sales class is responsible for

calculating budget of sales department and even the export revenue.

Step 3: Create a method named **exports()** in which all the export details are recorded.

The user is allowed to enter which product to be exported and in how much

quantity.

Step 4: Create another method **salarysales()** which is used to calculate the salary of n

employees using the method the **salarycalculator()** in the **Salarycalculate** class,

entered by the user.

Step 5: Create another method **calculate()**. This method is used to calculate or predict the

outcome of the project depending on the values entered by the user. It basically

takes the total salary budget into account and as well as the amount of revenue

made from exports.

Step 6: If the total budget of the company is lesser than what revenue it earns then

the company is said to be under profitable terms and can follow the same budget

plan.

Step 7: If the total budget of the company is greater than what revenue it earns then the company is said to be under be under loss and can might have to change the budgeting and exports plan which is recommended by the project too.

Step 8: End

**1.4 Feedback from the user:**

Step 1: START

Step 2: Using the I/O concepts, we take the feedback from the user regarding the quality

Of the product.

Step 3: This feedback is then recorded in to a specific file and the admin can have access

To view it.

Step 4: Only admin has the access to view the budget, export and feedback details.

Step 5: End

**CHAPTER 4**

**IMPLEMENTATION**

**IMPLEMENTATION OF JAVA CONCEPTS**

**Module 1 Functionality:**

****

**Fig.4.1 Class for calculating salary of all the department**

This module deals with the salary calculation of all the departments. Depending on the number of employments and basic salary either of the methods is used. We make use of the method overloading concept so that we can manage to calculate the salary of different department depending on their requirement. The first method with one parameter is used to calculate the salary of production department. Whereas the second method with taking two parameters is used to calculate the salary for administrative, customer service and sales department.

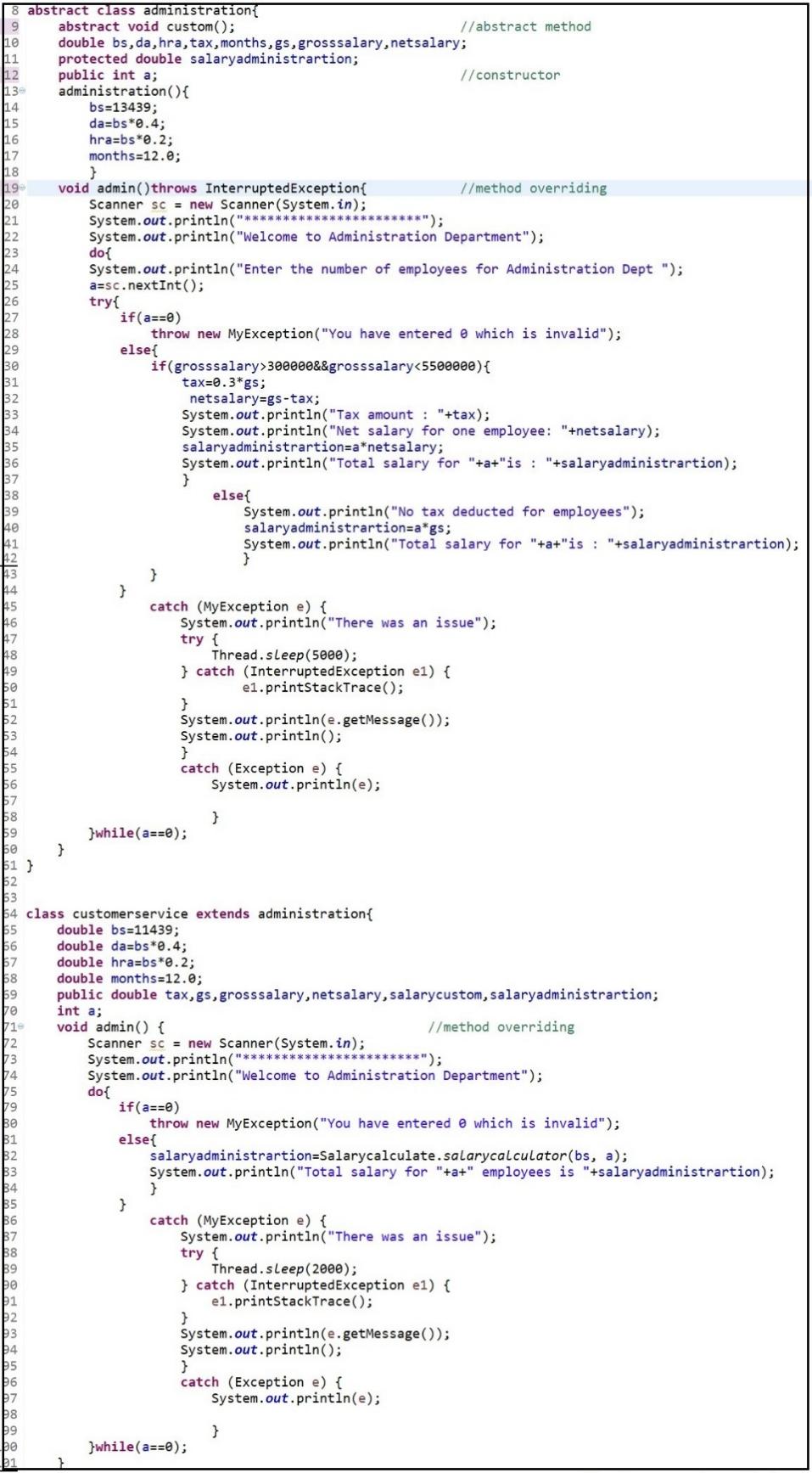
**Module 2 Functionality:**



**Fig.4.2 Customer Service class**

Customer service is one of the sectors of a company and it is provided with certain number of employees. This class is used to take input of number of employees and basic salary from the user and then call the salarycalculator() method to calculate total salary of the department and then return it back.

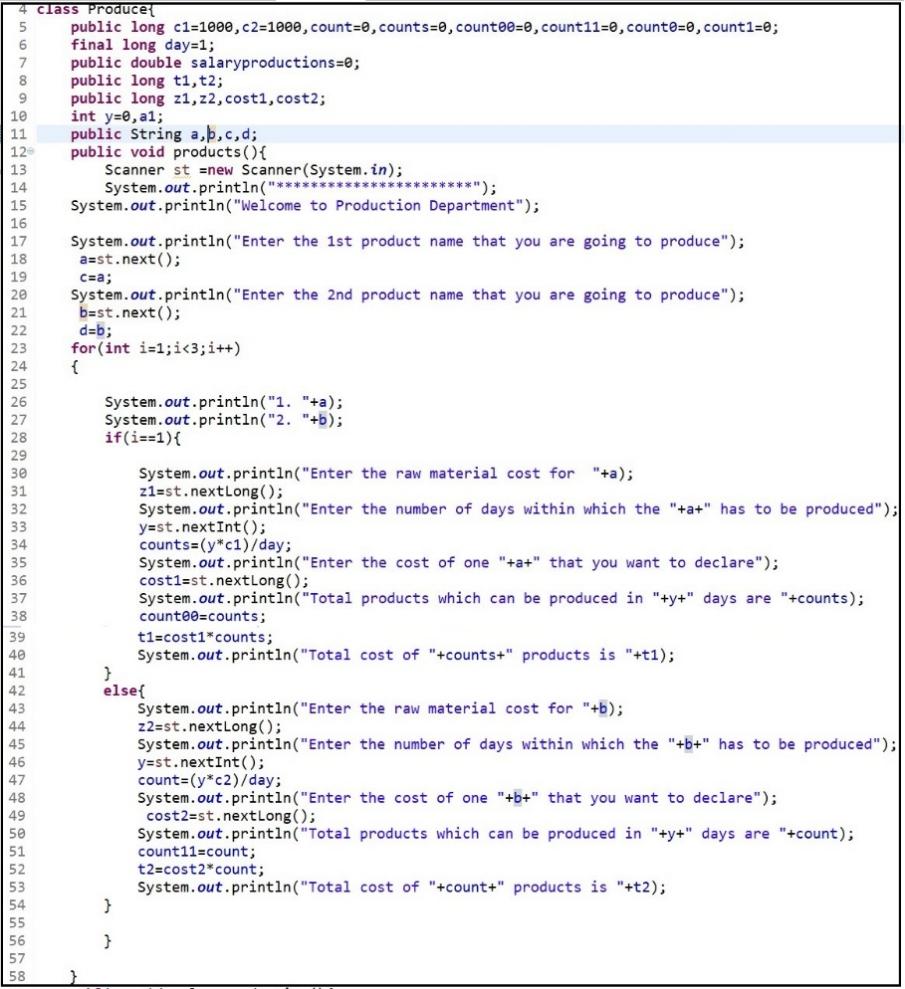
**Module 3 Functionality:**



**Fig.4.3 Overriding of administrative class**

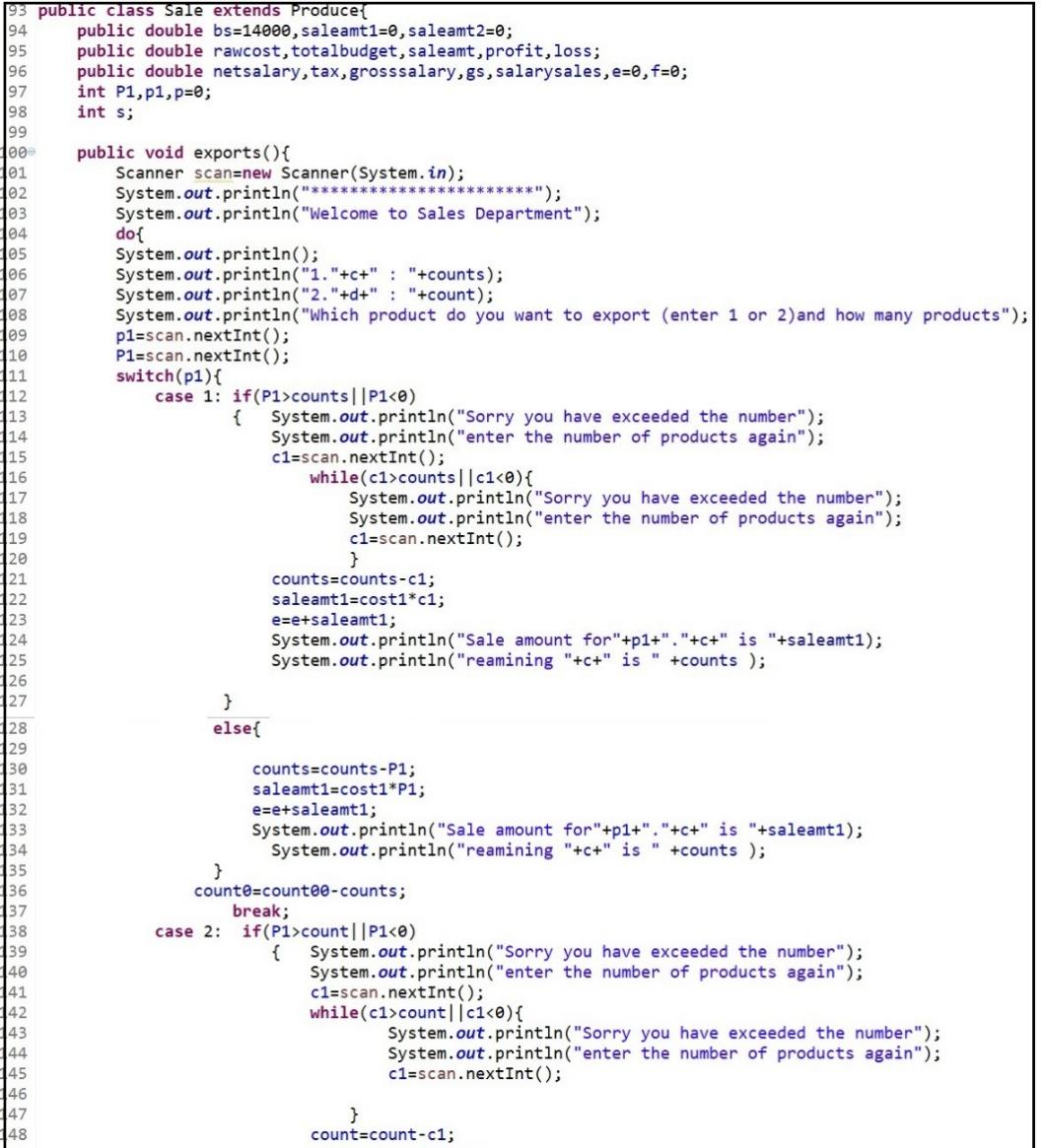
Administration is also one of the sectors of a company and is provided with certain number of employees. We have used the concept of method overriding so that we can override the parent class method when it doesn’t meet the criteria. In other words after one year if there needs to be a hike in the basic salary then we have to stop the override function by using final keyword to the child class method.

**Module 4 Functionality:**



**Fig.4.4.1 Class production**

The production department plays one of the vital role in terms of manufacturing products. So, this department also includes labours for which the salary has to be calculated and the procedure is same as followed by admin and customer service department. This class consist of method products() which the user has to enter two products name that has to be produced and their cost per product, raw material cost and number of days within which the product has to be produced respectively.

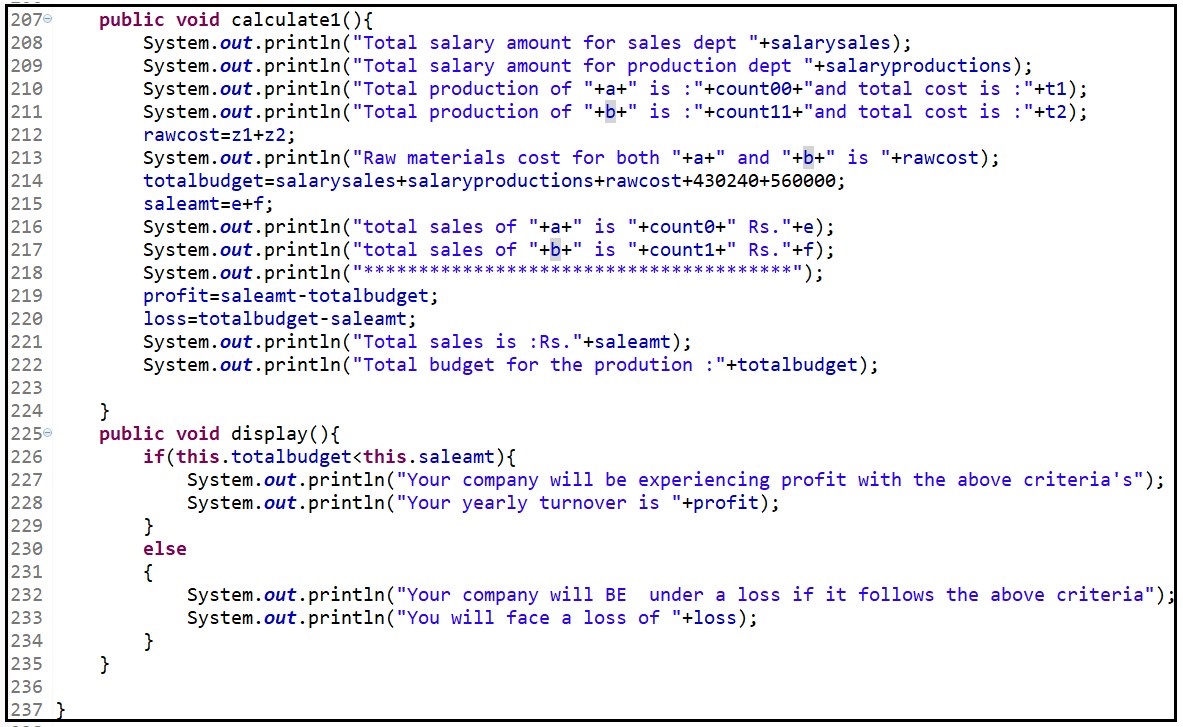


**Fig.4.4.2 Class sales (method export() )**

In the class sales we have the method export which gives user the option to export the products that were produced and in within the limit. If the quantity of the products has exhausted then it will move on to the budget and revenue calculation class.

The user can stop the export after a particular number too only that the left-over products will be considered damaged or malfunction and be discarded.

**Module 5 Functionality:**

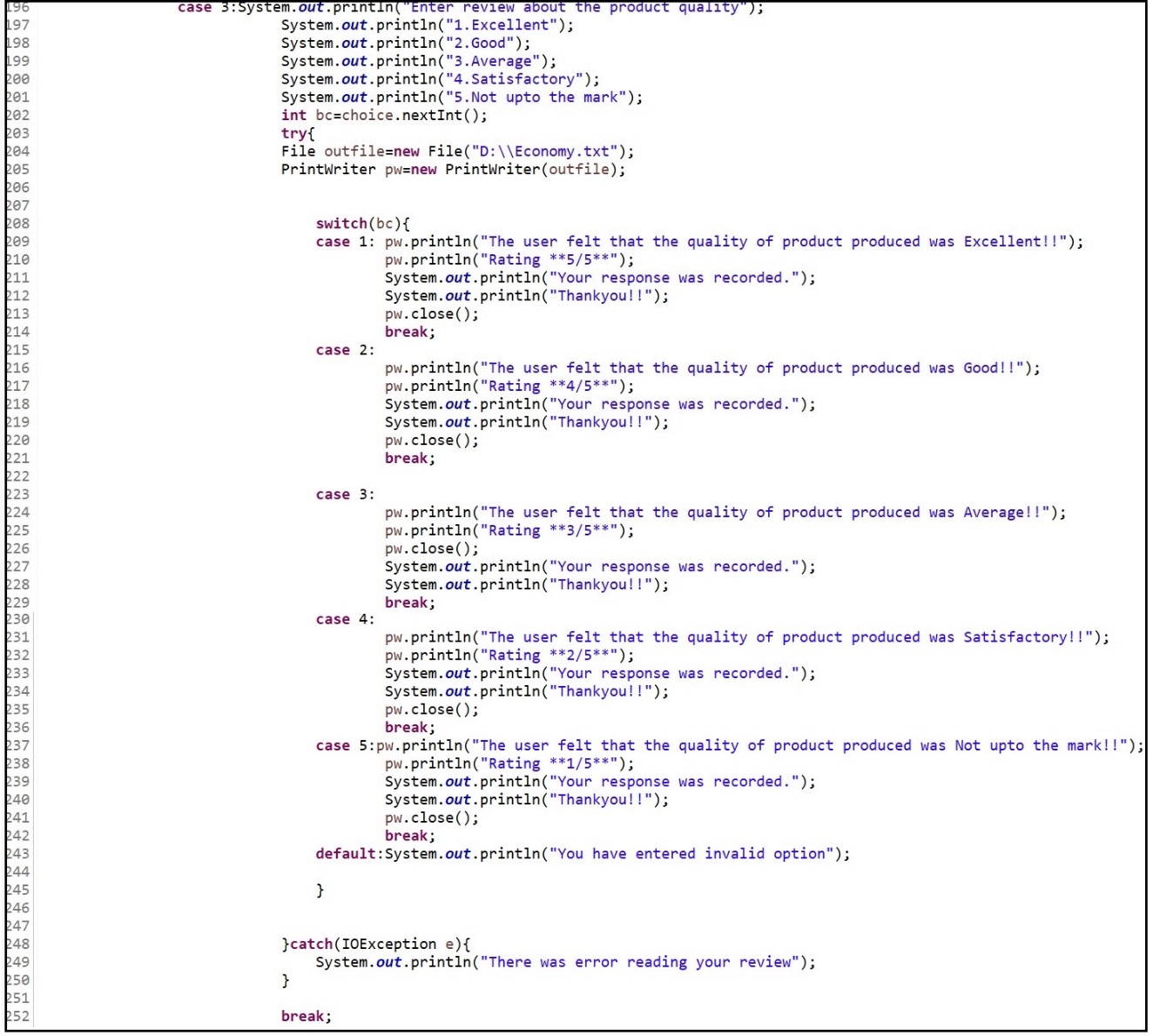


**Fig.4.5 Budget and revenue calculation in class sales**

This module explains the process calculating the budget of all the department in terms of salary and raw material cost. By summing up all the values given by each department the calculate() method calculates the total budget against the revenue earned by the company in terms of exporting the products.

If the total budget of the company is lesser than what revenue it earns then the company is said to be under profitable terms and can follow the same budget plan. If the total budget of the company is greater than what revenue it earns then the company is said to be under be under loss and can might have to change the budgeting and exports plan which is recommended by the project too.

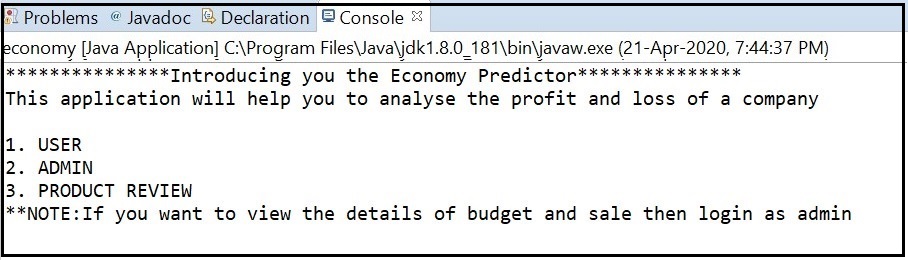
**Module 6 Functionality:**



**Fig.4.6 Feedback using I/O concepts**

This module explains about how the I/O concepts supported in java is used to take feedback from the user and then copy the same in a file which is mentioned in a particular directory given as the path name in the syntax of printwriter. All the budget details and feedbacks can be viewed only by the admin by which it provides a security to data without any threat.

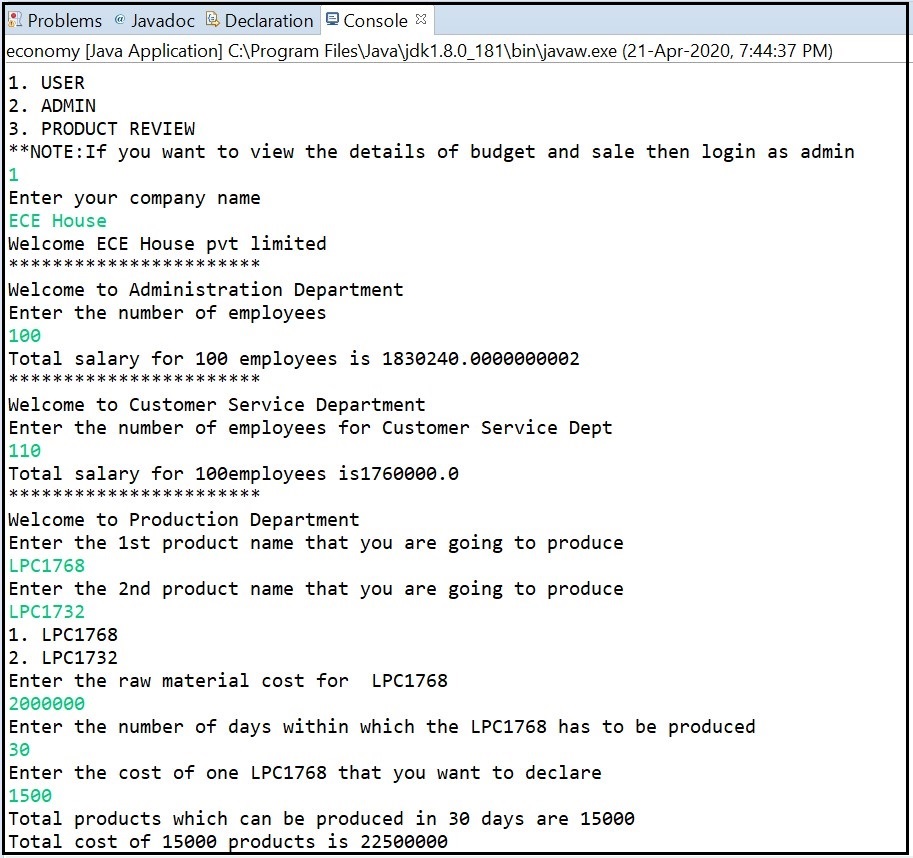
**5.RESULTS:**

****

**Fig.5.1 User – friendly menu**

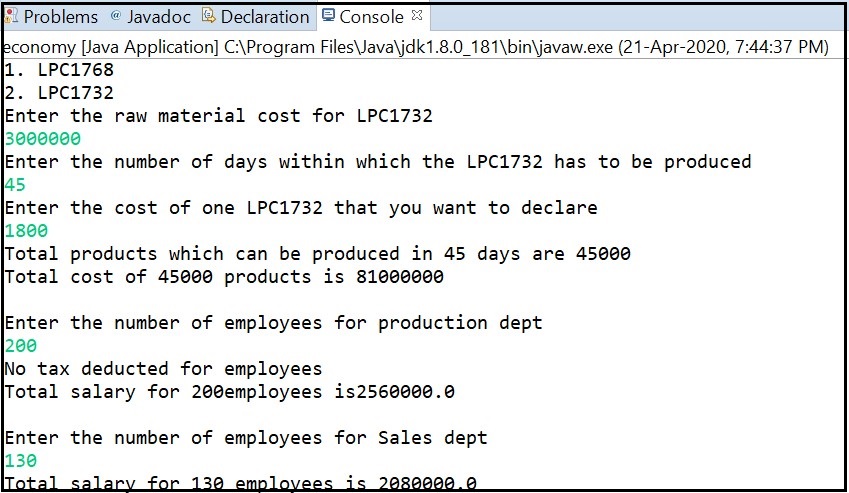
When application is compiled and executed successfully, it displays the user-friendly

menu consisting of options like user, admin and product review.



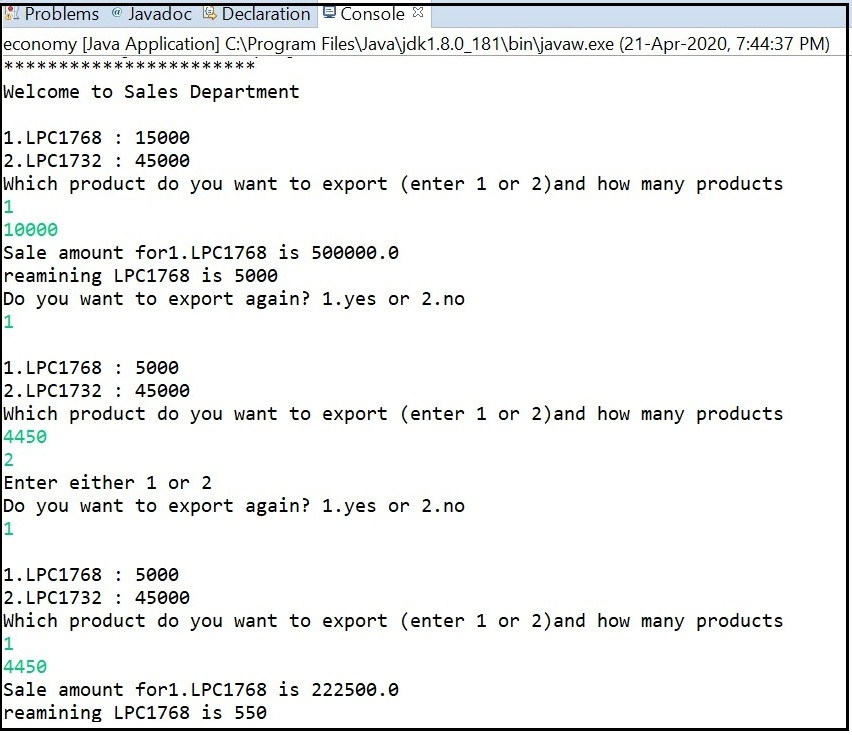
**Fig.5.2 Collecting details for admin and customer service.**

The above figure depicts the process of collecting data from the user in terms of number of employees in each of the departments. When coming to production department the user will have to enter the two products that has to be manufactured and also the raw material cost. The user will have to enter the number of days within which the product has to be produced. So, depending on the above data the program displays the quantity of products that can be produced in the specified number of days.

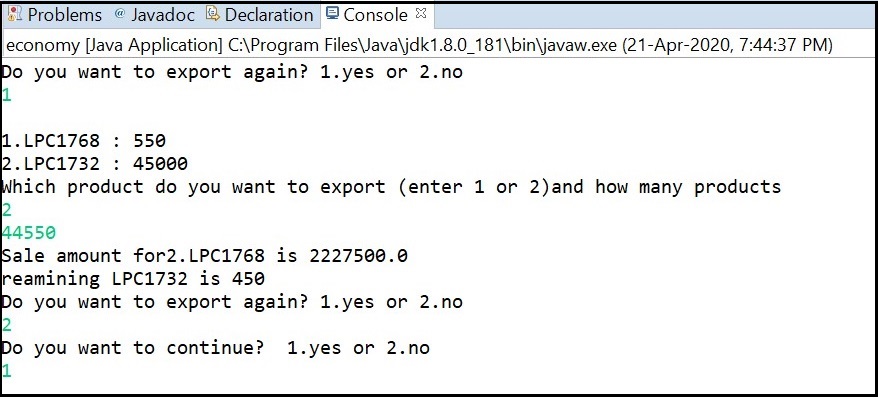
****

**Fig.5.3 Calculating salary for production department.**

This figure deals with the calculation of salary for the employees of production department. Tax is also calculated depending on their annual income which is, if the per annum income is less than 250000 then no tax is deducted from their income. If the annual income lies between 2500000 and 550000 then 15% of total income is deducted as tax and if the income greater than550000, then 30% of their total income is deducted as tax.

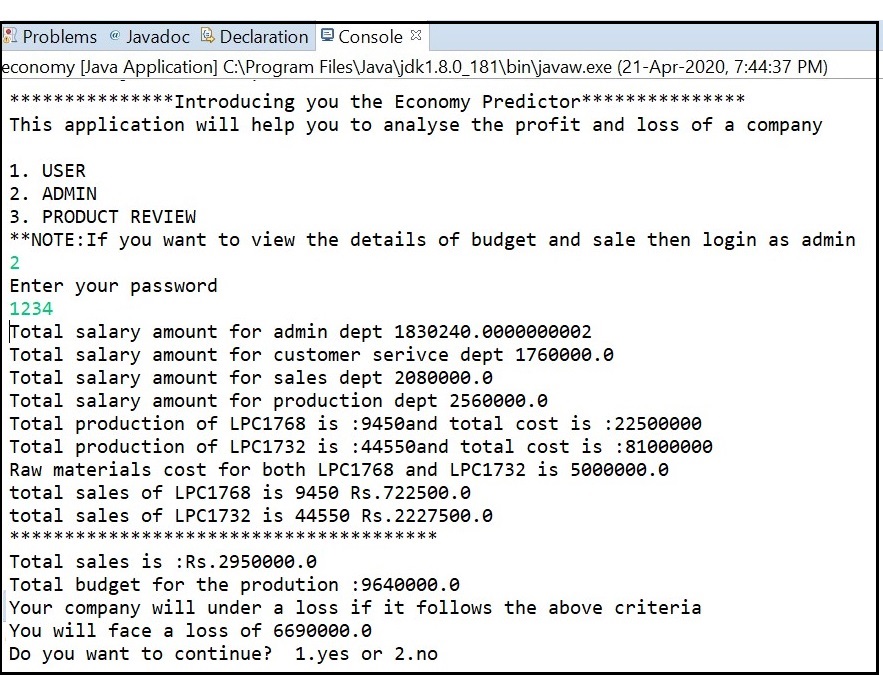


**Fig.5.4 Exports of product**

****

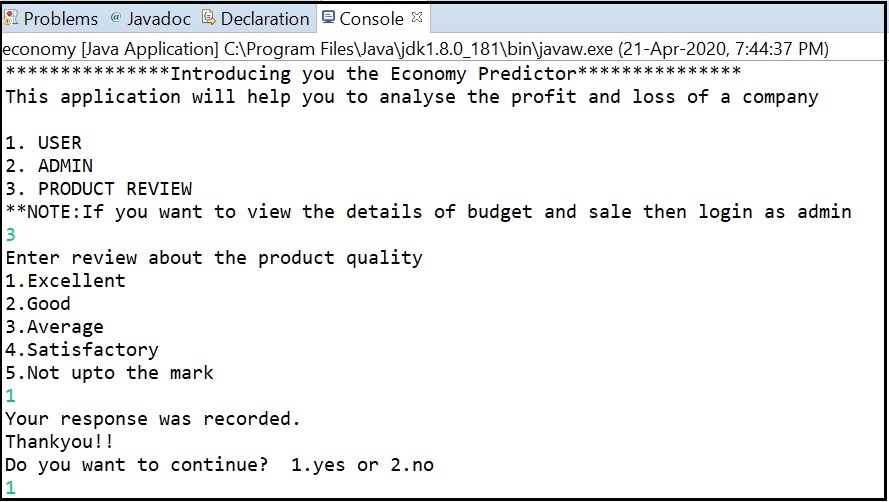
**Fig.5.5 Exports of product(continued)**

The above figures explains us about how the export of products takes place. It includes two products along with the quantity produced by the company. The user can export as many products as per the request and demand with the given limit. The total exports of the products leads to the revenue of the company.



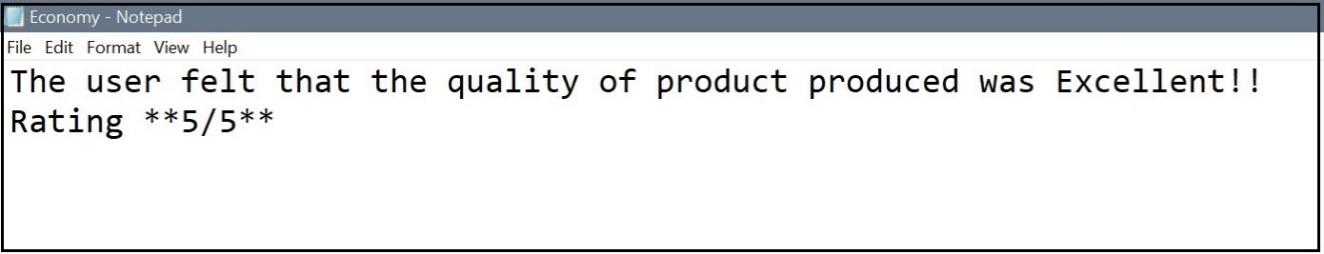
**Fig.5.6 Total Budget and revenue details**

Now comes the display section of the program which illustrates us about the total budgeting, expenses and revenue made by the company. This section decides the economic status of the company whether the company is in profitable terms are is it going under a loss.

****

**Fig.5.7 Feedback menu**

The user is allowed to give their opinion or feedback regarding the product and which is stored in a file. The feedbacks can be viewed only by the admin.

****

**Fig.5.8 File for storing the feedback from user.**

The above figure shows the feedback given by the user on the quality of the product which is in turn stored in a file which can be viewed by the admin anytime of convenience.

**6.CONCLUSION:**

EONOMY PREDICTOR application provides a very user-friendly interaction with the support of various java concepts. This application has various scopes in terms of determining if a company has to select a project depending on the outcomes produced by the project if it is performing well in terms of economy or not. The project can still be added with other features of database which will be useful to store the data for n companies and finally produce a statistic of which company among others is doing good in their economy. Finally, this project has helped me personally understand the object-oriented concepts very well and hence making this project more **efficient.**

**Hence Java programming language provide us several platforms to create real time projects.** **Therefore, this project relates to a real-world problem and provides a solution for it.**

**7. REFERENCES:**

The following websites are referred to create this project:

1.<https://www.javatpoint.com/java-io>

2. <https://www.javatpoint.com/java-oops-concepts>

3. <https://www.geeksforgeeks.org/java/>

4. <https://www.w3schools.com/java/java_oop.asp>

5. Herbert Schildt, Java™: The Complete Reference, McGraw-Hill, Tenth Edition,

2018