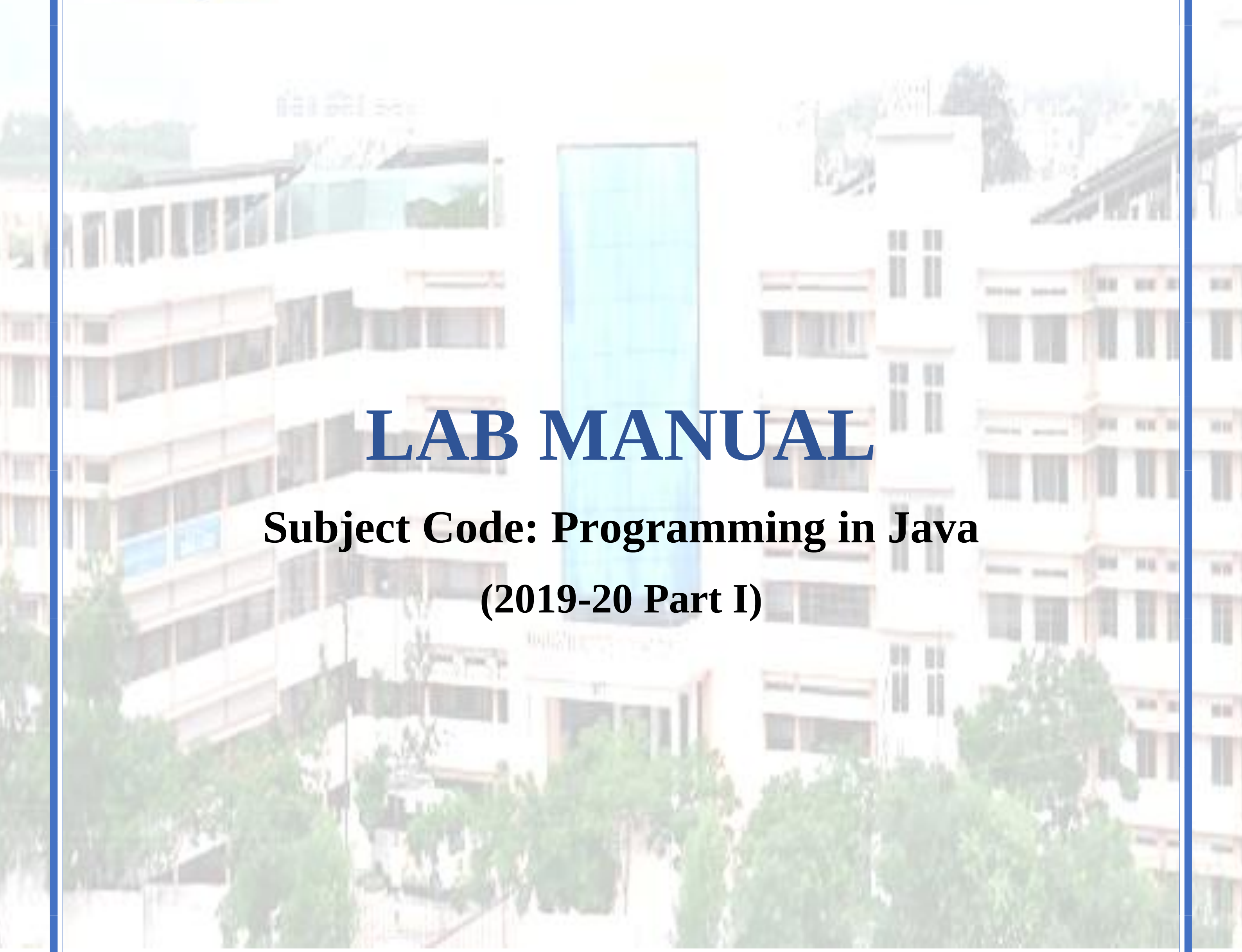
G.S. Mandal's

Maharashtra Institute of Technology, Aurangabad

Department of Computer Science and Engineering



LAB MANUAL

Subject Code: Programming in Java

(2019-20 Part I)

Maharashtra Institute of Technology, Aurangabad
NH-211, MIT Campus, Satara Village Road, Aurangabad- 431 010 (M.S.); India.
Phone: (0240) 2375222; Fax: (0240) 2376618, E-mail: principalmitt@mit.asia
Website: www.btech.mit.asia

Department of Computer Science and Engineering

Vision

To develop the department as a center of excellence in the field of computer science and engineering by imparting knowledge & training to the students for meeting growing needs of the industry & society.

Mission

Providing quality education through a well-designed curriculum in tune with the challenging needs of software industry by providing state of the art facilities and to impart knowledge in the thrust areas of computer science and engineering.

Department of Computer Science and Engineering

Program Educational Objectives

PEO1: To prepare the students to achieve success in Computing Domain to create individual careers, innovations or to work as a key contributor to the private or Government sector and society.

PEO2: To develop the ability among the students to understand Computing and mathematical fundamentals and apply the principles of Computer Science for analyzing, designing and testing software for solving problems.

PEO3: To empower the students with ability to quickly reflect the changes in the new technologies in the area of computer software, hardware, networking and database management.

PEO4: To promote the students with awareness for lifelong learning, introduce them to professional practice, ethics and code of professionalism to remain continuous in their profession and leaders in technological society.

Program Specific Objectives

PSO1: Identify appropriate data structures and algorithms for a given contextual problem and develop programs to design and implement web applications.

PSO3: Design and manage the large databases and develop their own databases to solve real world problems and to design, build, manage networks and apply wireless techniques in mobile based applications.

PSO3: Design a variety of computer-based components and systems using computer hardware, system software, systems integration process and use standard testing tools for assuring the software quality.

Program Outcomes

PO1: Apply knowledge of mathematics, science, and engineering fundamentals to solve problems in Computer science and Engineering.

PO2: Identify, formulate and analyze complex problems.

PO3: Design system components or processes to meet the desired needs within realistic constraints for the public health and safety, cultural, societal and environmental considerations.

PO4: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data for valid conclusions.

PO5: Select and apply modern engineering tools to solve the complex engineering problem.

PO6: Apply knowledge to assess contemporary issues.

PO7: Understand the impact of engineering solutions in a global, economic, environmental, and societal context.

PO8: Apply ethical principles and commit to professional ethics and responsibilities.

PO9: Work effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

PO10: Communicate effectively in both verbal and written form.

PO11: Demonstrate knowledge and apply engineering and management principles to manage projects and in multi-disciplinary environment.

PO12: To engage in life-long learning to adopt to the technological changes.

Department of Computer Science and Engineering

Course: 323: Programming in Java

Course Outcomes:

After Completing the course students will be able to

CO1 : Apply the concepts of data abstraction, encapsulation, polymorphism, and inheritance for problem solving in Java.

CO2 : Implement the concepts of packages and interfaces

CO3 : Develop programs for exception handling and multi threading.

CO4 : Demonstrate applets and IO application programs in Java.

CO5 : Design GUI applications using swing components

CO6 : Perform basic sql operation on database with the help of Java

Mapping

Experiment No.	Blooms Level	Mapping To CO	Mapping To PO
1	3	CO1	PO1,PO3,PO5
2	3	CO1	PO1,PO3,PO5
3	3	CO1	PO1,PO2,PO3,PO5
4	3	CO2	PO1,PO3,PO5
5	3	CO3	PO1,PO3,PO5
6	3	CO3	PO1,PO3,PO5
7	3	CO4	PO3,PO5
8	3	CO4	PO3,PO5
9	3	CO5	PO2,PO3,PO5
10	3	CO6	PO2,PO3,PO5

Index

	divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 10. b). Problem Statement Create an interface Stack in package myPack with following methods Methods: void push(int) int pop Create two subclasses inheriting the Stack interface. class FixedStack that can accept a fixed numbers of integers. The size of the Stack will be specified in the constructor. If the number of integers exceeds the size, "Stack overflow" message should be displayed class VariableStack that can accept numbers more than its size. The size of the Stack will be specified in the constructor. If the number of integers exceeds the size, the stack should automatically grow to accommodate the new element.	
5	1)If user enters minus number throw NegativeNumberException which is a user defined exception. 2) Create BankAccount with 500 Rs Minimum Balance, Deposit Amount, Withdraw Amount and Also Throws LessBalanceException. - Class Called LessBalanceException Which returns the Statement that Says WithDraw Amount(_Rs) is Not Valid. Write a Java Program that has a Class Which Creates Account, perform Deposite Money and Tries to WithDraw more Money Which Generates a LessBalanceException. Take Appropriate Action for the Same	**22**
6	a) Write a program to create two child threads extending Thread class b) Write a program to create two child threads implementing Runnable class c) Write a program demonstrate use of sleep, join and isAlive functions d) Write a program to demonstrate synchronization on common resources	**23**
7	a) Write a program to copy contents of a file using IO stream b) Write a program to encrypt and decrypt file contents c) Write a program to read and write data using Random Access File d) Write a program to save state an object(object serialization /deserialization)	**24**
8	Write a program to demonstrate Applet life cycle	**25**
9	a) Write a program to design login form and check valid user b) Write a program to design calculator and perform simple arithmetic operation c) write a program to design student college form (Refer image 1) d) Write a program for java online test (Minimum 3 questions). Display score e) Write a program to handle mouse listener	**26-27**

	f) Write a program to create File, Edit menu. File menu have new, open, close and edit menu with cut, copy, paste options	
10	Mysql Database : Organization , Table : Dept(id int, dname varchar(02), lab int) a) Write a program to insert ,delete, update and select records using statement b) Write a program to insert ,delete, update and select records using PreparedStatement Take input from user.	**28**

Appendix A

Appendix B

G.S. MANDAL’S

MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD

LAB WORK INSTRUCTION SHEET

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CLASS: T.Y. B. TECH PART: 1 (2019-20)	**LAB:**	**SUBJECT: CSE303-Programming in Java**

Experiment Number : 01

Aim: To demonstrate object oriented programming in java

Objective: To understand object oriented programming in java

Title: Write a class Stack(or /Queue/Arithmetic operations) to push number, pop number and also to check stack empty or full.

Outcomes: Students will able to apply object oriented programming in java

Theory:

class in Java

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- **Fields**
- **Methods**
- **Constructors**
- **Blocks**
- **Nested class and interface**

Objects

An object in Java is the physical as well as a logical entity, whereas, a class in Java is a logical entity only.

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

An object has three characteristics:

- **State:** represents the data (value) of an object.

- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object Definitions:

- An object is *a real-world entity*.
- An object is *a runtime entity*.
- The object is *an entity which has state and behavior*.
- The object is *an instance of a class*.

Code to create Object in java –
<ClassName> objectName=new <ClassName>
For Example :

Employee e1=new Employee();

Constructor :

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.
It is a special type of method which is used to initialize the object.
Every time an object is created using the new() keyword, at least one constructor is called.
It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

- A Constructor must have no explicit return type
- A Java constructor cannot be abstract, static, final, and synchronized
-

There are two types of constructors in Java: no-arg constructor, and parameterized constructor.
Example of default constructor

```
//Java Program to create and call a default constructor
class Bike1{
```

```
//creating a default constructor
Bike1(){System.out.println("Bike is created");}
//main method
public static void main(String args[]){
//calling a default constructor
Bike1 b=new Bike1();
}  }
```

Access specifiers in Java

There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Access Specifiers	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Java Naming conventions

Java naming convention is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method, etc.

But, it is not forced to follow. So, it is known as convention not rule. These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.

All the classes, interfaces, packages, methods and fields of Java programming language are given according to the Java naming convention. If you fail to follow these conventions, it may generate confusion or erroneous code.

Advantage of naming conventions in java

By using standard Java naming conventions, you make your code easier to read for yourself and other programmers. Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

The following are the key rules that must be followed by every identifier:

- The name must not contain any white spaces.
- The name should not start with special characters like & (ampersand), $ (dollar), _ (underscore).

Let's see some other rules that should be followed by identifiers.

Class

- It should start with the uppercase letter.
- It should be a noun such as Color, Button, System, Thread, etc.
- Use appropriate words, instead of acronyms.
- **Example: -**

```
public class Employee
{
```

```
//code snippet
}
```

Interface

- It should start with the uppercase letter.
- It should be an adjective such as Runnable, Remote, ActionListener.
- Use appropriate words, instead of acronyms.
- **Example: -**

```
interface Printable
{
//code snippet
}
```

Method

- It should start with lowercase letter.
- It should be a verb such as main(), print(), println().
- If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().
-
-

Example:-

```
 class Employee
{
//method
void draw()
{
//code snippet
}
}
```

Variable

- It should start with a lowercase letter such as id, name.
- It should not start with the special characters like & (ampersand), $ (dollar), _ (underscore).
- If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName.
- Avoid using one-character variables such as x, y, z.
- **Example :-**

```
 class Employee
{
//variable
int id;
//code snippet
}
```

Package

- It should be a lowercase letter such as java, lang.
- If the name contains multiple words, it should be separated by dots (.) such as java.util, java.lang.
-
-

Example :-

```
package com.javatpoint; //package
class Employee
{
//code snippet
}
```

Constant

- It should be in uppercase letters such as RED, YELLOW.
- If the name contains multiple words, it should be separated by an underscore(_) such as MAX_PRIORITY.

- It may contain digits but not as the first letter.
- **Example :-**

```
class Employee
{
//constant
 static final int MIN_AGE = 18;
//code snippet
}
```

CamelCase in java naming conventions

Java follows camel-case syntax for naming the class, interface, method, and variable.

If the name is combined with two words, the second word will start with uppercase letter always such as actionPerformed(), firstName, ActionEvent, ActionListener, etc.

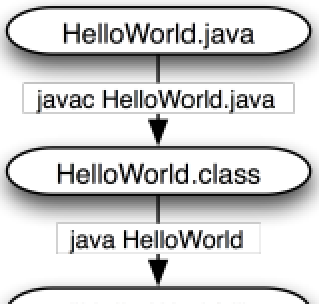
Compiling Java Programs

- The JDK comes with a command-line compiler: **javac**.
  - It compiles source code into *Java bytecode*, which is low-level instruction set similar to binary machine code.
  - The bytecode is executed by a *Java virtual machine* (*JVM*), rather than a specific physical processor.
- To compile our **HelloWorld.java**, you could go to the directory containing the source file and execute:

```
javac HelloWorld.java
```

- This produces the file **HelloWorld.class**, which contains the Java bytecode.

Use the **javac -help** option to get a full list of available options:



Running Java Programs

- To run the bytecode, execute:

```
java HelloWorld
```

- Do not include the **.class** extension.
- The **java** command starts the JVM and executes the class bytecode.
- The JVM abstracts O/S and H/W from the Java application.

It is the Java virtual machine that provides the layer of insulation to Java programs so that they do not depend on the underlying operating system or hardware semantics.

public static void main(String[] args)

Java main method is the entry point of any java program. Its syntax is always public static void main(String[] args). You can only change the name of String array argument

Stack Operations –

A stack is an Abstract Data Type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack, for example – a deck of cards or a pile of plates, etc.

A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, we can only access the top element of a stack.

This feature makes it LIFO data structure. LIFO stands for Last-in-first-out.

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations −

- **push()** − Pushing (storing) an element on the stack.
- **pop()** − Removing (accessing) an element from the stack.

When data is PUSHed onto stack.

To use a stack efficiently, we need to check the status of stack as well. For the same purpose, the following functionality is added to stacks −

- **peek()** − get the top data element of the stack, without removing it.
- **isFull()** − check if stack is full.
- **isEmpty()** − check if stack is empty.

At all times, we maintain a pointer to the last PUSHed data on the stack. As this pointer always represents the top of the stack, hence named **top**. The **top**pointer provides top value of the stack without actually removing it.

Courtesy :
https://www.javatpoint.com
https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm

Source Code:

Output: <screen shots>

Conclusion : (note: write in your own handwriting what u have studied from this experiment)
I have studied……

1) **To use/apply ____ ______________________ _____________________ _________ ____ _________**

   _______________________in java

2) **List Errors you have faced during execution of program :**

   --

CLASS: T.Y. B. TECH PART: 1 (2019-20)	**LAB:**	**SUBJECT: CSE303-Programming in Java**

Experiment Number : 02
Aim: To demonstrate String , ArralyList,Vector,HashMap collection in java

Objective: To understand String, collection in java

Title:
a) Write a Java program to test if a given string contains the specified sequence of char values.
b) Write a Java program to search color in given ArrayList of colors.
c) Write a Java program to search if a hashmap contains a mapping for the specified key, specified value.

Outcomes: Students will able to apply string functions, access collection in java

Theory:

Strings

Strings, which are widely used in Java programming, are a sequence of characters. In the Java programming language, strings are objects. The Java platform provides the String class to create and manipulate strings.

Creating Strings

char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
String helloString = new String(helloArray);
The most direct way to create a string is to write:
String greeting = "Hello world!";

Some Java String class methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

char charAt(int index) - returns char value for the particular index

int length()- returns string length

String substring(int beginIndex) -returns substring for given begin index.

String substring(int beginIndex, int endIndex)- returns substring for given begin index and end index.

boolean contains(CharSequence s)-returns true or false after matching the sequence of char value.

boolean equals(Object another)-checks the equality of string with the given object.

String concat(String str) - concatenates the specified string.

static String equalsIgnoreCase(String another) -compares another string. It doesn't check case.

String toLowerCase() - returns a string in lowercase.

String toUpperCase() - returns a string in uppercase.

Converting Numbers to Strings :

```
int i=7;
double d=4;
String s3 = Integer.toString(i);
String s4 = Double.toString(d);
// The valueOf class method.
String s2 = String.valueOf(i);
```

Converting String to Numbers :

```
String s= “5”;
int a=Integer.parseInt(s);
double b=Double.parseDouble(s);
```

Note : The Integer,Double class wraps a value of the primitive type in an object

Introduction to Collections

A collection is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. Typically, they represent data items that form a natural group, a mail folder (a collection of letters), or a telephone directory (a mapping of names to phone numbers).

What Is a Collections Framework?

A *collections framework* is a unified architecture for representing and manipulating collections. All collections frameworks contain the following:

- **Interfaces:** These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.
- **Implementations:** These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.
- **Algorithms:** These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces.

The Java Collections Framework provides the following benefits:

- **Reduces programming effort:** By providing useful data structures and algorithms, the Collections Framework frees you to concentrate on the important parts of your program rather than on the low-level "plumbing" required to make it work.
- **Increases program speed and quality:** This Collections Framework provides high-performance, high-quality implementations of useful data structures and algorithms..

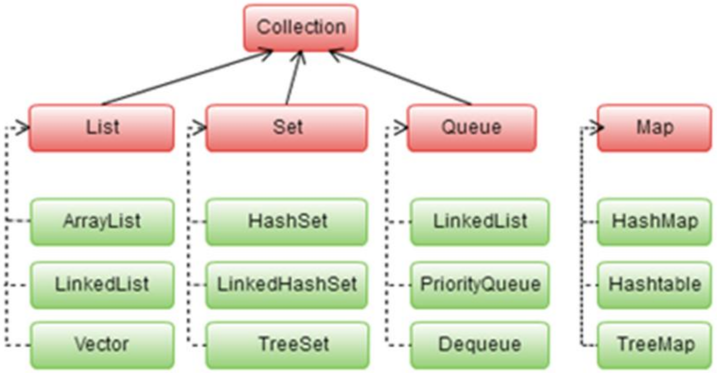


Figure 2.1: Java Collection Framework

- Collection — the root of the collection hierarchy. A collection represents a group of objects known as its *elements*. The Collection interface is the least common denominator that all collections implement and is used to pass collections around and to manipulate them when maximum generality is desired. Some types of collections allow duplicate elements, and others do not. Some are ordered and others are unordered. The Java platform doesn't provide any direct implementations of this interface but provides implementations of more specific subinterfaces, such as Set and List.

- Set — a collection that cannot contain duplicate elements. This interface models the mathematical set abstraction and is used to represent sets, such as the cards comprising a poker hand, the courses making up a student's schedule, or the processes running on a machine.

- List — an ordered collection (sometimes called a *sequence*). Lists can contain duplicate elements. The user of a List generally has precise control over where in the list each element is inserted and can access elements by their integer index (position).
- Queue — a collection used to hold multiple elements prior to processing. Besides basic Collection operations, a Queue provides additional insertion, extraction, and inspection operations.

  Queues typically, but do not necessarily, order elements in a FIFO (first-in, first-out) manner. Among the exceptions are priority queues, which order elements according to a supplied comparator or the elements' natural ordering. Whatever the ordering used, the head of the queue is the element that would be removed by a call to remove or poll. In a FIFO queue, all new elements are inserted at the tail of the queue. Other kinds of queues may use different placement rules. Every Queueimplementation must specify its ordering properties..

- Deque — a collection used to hold multiple elements prior to processing. Besides basic Collection operations, a Dequeprovides additional insertion, extraction, and inspection operations.

  In a deque all new elements can be inserted, retrieved and removed at both ends..

- Map — an object that maps keys to values. A Map cannot contain duplicate keys; each key can map to at most one value. If you've used Hashtable, you're already familiar with the basics of Map.

Traversing Collections

There are three ways to traverse collections: (1) using aggregate operations (2) with the for-each construct and (3) by using Iterators.

In JDK 8 and later, the preferred method of iterating over a collection is to obtain a stream and perform aggregate operations on it. Aggregate operations are often used in conjunction with lambda expressions to make programming more expressive, using less lines of code. The following code sequentially iterates through a collection of shapes and prints out the red objects:

```
myShapesCollection.stream()

.filter(e -> e.getColor() == Color.RED)
```

```
.forEach(e -> System.out.println(e.getName()));
```

for-each Construct :

The for-each construct allows you to concisely traverse a collection or array using a for loop — see The for Statement. The following code uses the for-each construct to print out each element of a collection on a separate line.

```
for (Object o : collection)

   System.out.println(o);
```

Iterators :

An Iterator is an object that enables you to traverse through a collection and to remove elements from the collection selectively, if desired. You get an Iterator for a collection by calling its iterator method.

The hasNext method returns true if the iteration has more elements, and the next method returns the next element in the iteration. The remove method removes the last element that was returned by next from the underlying Collection. The remove method may be called only once per call to next and throws an exception if this rule is violated

Package require : java.util

ArrayList

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed.

Example -

```
        ArrayList<String> list=new ArrayList<String>();//Creating arraylist
        list.add("Ravi");//Adding object in arraylist
```

Vector

Vector uses a dynamic array to store the data elements. It is similar to ArrayList. However, It is synchronized and contains many methods that are not the part of Collection framework.

Example –

```
Vector<String> v=new Vector<String>();
v.add("Ayush");
```

HashMap

HashMap class implements the map interface by using a hash table. It inherits AbstractMap class and implements Map interface. Java HashMap class contains values based on the key.Java HashMap class contains only unique keys.Java HashMap class may have one null key and multiple null values.Java HashMap class is non synchronized.Java HashMap class maintains no order.

Example –

```
HashMap<Integer,String> map=new HashMap<Integer,String>();
     map.put(100,"AAA");
   //key-based removal
   map.remove(100);
```

Source Code:

Output: <screen shots>

Conclusion : (note: write in your own handwriting what u have studied from this experiment)

I have applied

- **string methods ____ ____________________ ____________________ _________ ____ _________**

  _____________________in java
- **Collection methods ____________ ___________________ ________**

- **List Errors you have faced during execution of program -------------------------------------**

G.S. MANDAL’S MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD		
LAB WORK INSTRUCTION SHEET		
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING		
CLASS: T.Y. B. TECH PART: 1 (2019-20)	**LAB:**	**SUBJECT: CSE303-Programming in Java**

Experiment Number : 03

Aim: To demonstrate relationship among classes

Objective: To understand inheritance(is-a), has-a relationship

Title:

a) In a library there are two kinds of users- issuer ,student,faculty and administrator, who plays different roles like suggesting books, issuing a book, checking and updating the inventory, checking the dues, etc.

b) The payroll system of an organization keeps track of salaries of employees; the employee may be a permanent employee, daily wager, as per their status they will get bonus and increments.

c) Bank want to automate the process of managing their accounts- the bank provides following facilities: Opening a new account, Deposit amount,Withdraw amount,View the account details.

The account holder has to provide with following information while opening account:account number,account type- savings, current, Account holders name, address, current balance etc

Outcomes: Students will able to identify classes and relationship, implement program for same.

Theory:

Inheritance in Java

It is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of Object Oriented programming.

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

Example :

```
class  Superclass-name { .. }

class Subclass-name extends Superclass-name      //  extends keyword use to inherit super class
{
  //methods and fields
}
```

Types of Inheritance in Java

Based on class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only.

Single Inheritance :

In single inheritance, subclasses inherit the features of one superclass. In image below, the class A serves as a super class for the sub class B.

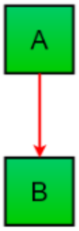


Figure 3.1 : Single Inheritance

Multilevel Inheritance :

In Multilevel Inheritance, a sub class will be inheriting a super class and as well as the sub class also act as the super class to other class. In below image, the class A serves as a super class for the sub class B, which in turn serves as a super class for the sub class C. In Java, a class cannot directly access the grandparent’s members.

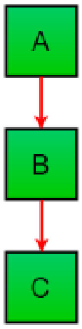


Figure 3.2 : Multilevel Inheritance

Hierarchical Inheritance :

In Hierarchical Inheritance, one class serves as a superclass for more than one sub class. In below image, the class A serves as a super class for the sub class B,C and D.

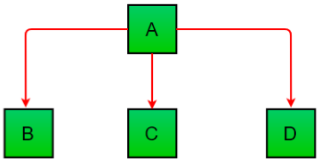


Figure 3.3 : Hierarchical Inheritance

Multiple Inheritance (Through Interfaces) :

In Multiple inheritance ,one class can have more than one superclass and inherit features from all parent classes. Java does **not** support multiple inheritance with classes. In java, we can achieve multiple inheritance only through Interfaces.

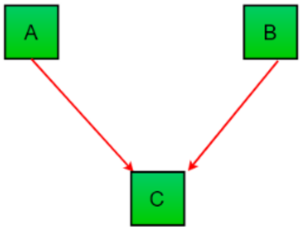


Figure 3.4 : Hierarchical Inheritance

Hybrid Inheritance(Through Interfaces) :

It is a mix of two or more of the above types of inheritance. Since java doesn’t support multiple inheritance with classes, the hybrid inheritance is also not possible with classes. In java, we can achieve hybrid inheritance only through Interfaces.

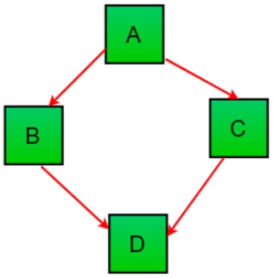


Figure 3.5 : Hybrid Inheritance

A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass by using **super** keyword.

Aggregation in Java (has-a relation)

If a class have an entity reference, it is known as Aggregation. Aggregation represents HAS-A relationship.

Consider a situation, Employee object contains many informations such as id, name, emailId etc. It contains one more object named address, which contains its own informations such as city, state, country, zipcode etc. as given below.

```
class Employee{
int id;
String name;
Address address;//Address is a class
...
}
```

In such case, Employee has an entity reference address, so relationship is Employee HAS-A address.

For example :

```
public class Emp {
int id;
String name;
Address address;

public Emp(int id, String name,Address address) {
    this.id = id;
    this.name = name;
    this.address=address;
. }
. void display(){
. System.out.println(id+" "+name);
. System.out.println(address.city+" "+address.state+" "+address.country);
. }
.
public static void main(String[] args) {
Address address1=new Address("abad","MS","india");
Address address2=new Address("pune","MS","india");

Emp e1=new Emp(111,"ABC",address1);
Emp e2=new Emp(112,"arun",address2);

e1.display();
e2.display();

}
}
```

Source Code:

Output: <screen shots>

Conclusion : (note: write in your own handwriting what u havee studied from this experiment)
I have applied

- **Type of inheritance** ____ ____________________ ___________________ ________ ____ ________

  ____

- **Any has – a relation ship :** ___________ __________________ _______

- **List Errors you have faced during execution of program :**

Experiment Number : 04
Aim : To demonstrate package, interface
Objective : To understand and apply package,interface in java

Problem Statement :
1.Write an interface AdvancedArithmetic which contains a method signature int divisor_sum(int n). You need to write a class called MyCalculator which implements the interface.
divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 10.
2. Problem Statement Create an interface Stack in package myPack with following methods Methods: void push(int) int pop Create two subclasses inheriting the Stack interface. class FixedStack that can accept a fixed numbers of integers. The size of the Stack will be specified in the constructor. If the number of integers exceeds the size, "Stack overflow" message should be displayed class VariableStack that can accept numbers more than its size. The size of the Stack will be specified in the constructor. If the number of integers exceeds the size, the stack should automatically grow to accommodate the new element.

Outcome : Students will able to apply interface in java
Theory :
(Describe following points . Note : wherever applicable write java syntax **)**
Package :
Create package –

Access package –
Java.lang package-

interface –

data members and methods in interface –

implements keyword –
multiple inheritance using interface-
difference between abstract class and interface -

Java Code :
Output: ***(note: add screen shots)***
Conclusion :
I have studied……

1) **To create and access package**
2) **To ….. interface**
3) **About interface methods…..**

Experiment Number : 05

Aim : To study user defined exception

Objective : To understand and apply user defined exception in java

Problem Statement :

1)If user enters minus number throw NegativeNumberException which is a user defined exception.

2) Create BankAccount with 500 Rs Minimum Balance, Deposit Amount, Withdraw Amount and Also Throws LessBalanceException.

- Class Called LessBalanceException Which returns the Statement that Says WithDraw Amount(_Rs) is Not Valid. Write a Java Program that has a Class Which Creates Account, perform Deposite Money and Tries to WithDraw more Money Which Generates a LessBalanceException. Take Appropriate Action for the Same

Outcome : Students will able to apply user defined exception in java

Theory :

(Describe following points. Note : wherever applicable write java syntax **)**

Exception class hierarchy -

Use of try-catch-finally

throw –

throws -

user defined exception –

How to define it

Java Code :

Output: ***(note: add screen shots)***

Conclusion :

I have studied……

1) **I have defined user defined exception ………..**
2) **I have handled exception using ….**
3) **By using …… I have thrown user defined exception**

Experiment Number : 06
Aim : To use multithreading in java
Objective : To understand and apply multithreading in java
Problem Statement :

a) Write a program to create two child threads extending Thread class
b) Write a program to create two child threads implementing Runnable class
c) Write a program demonstrate use of sleep, join and isAlive functions
d) Write a program to demonstrate synchronization on common resources

Outcome : Students will able to apply multithreading in java
Theory :
***(Note : wherever applicable write java syntax*)**
Thread and Runnable –
Methods start,run,sleep,join,isAlive, get, setPriority
Synchronization –

Java Code :
Output: ***(note: add screen shots)***

Conclusion :
I have studied……

1) **I have applied multithreading using ……….. class and …… interface**
2) **These exceptions are thrown …. (list of exceptions) in multithreading**
3) **In program , without synchronization ………… ……**
   After applying synchronization …….

Experiment Number : 07
Aim : To use IO in java
Objective : To understand and apply file IO in java
Problem Statement :

a) Write a program to copy contents of a file using IO stream
b) Write a program to encrypt and decrypt file contents
c) Write a program to read and write data using Random Access File
d) Write a program to save state an object(object serialization /deserialization)

Outcome : Students will able to apply IO in java
Theory :
(Note : wherever applicable write java syntax)
InputStrea /OutputStream Hierarchy –
Reader/Write Hierarchy
Use of BufferedReader –
Input from keyboard using BufferedReader -
Random Access File –
Serialization and Deserialization

Java Code :
Output: ***(note: add screen shots)***

Conclusion :
I have studied……

1) **These exceptions are thrown …. (list of exceptions) in IO**
2) **Advantage of Random Access File**
3) **By using serialization …..**

Experiment Number : 08
Aim : To study Applet
Objective : To understand and apply Applet in java
Problem Statement: Write a program to demonstrate Applet life cycle

Outcome : Students will able to understand Applet life cycle in java
Theory :
(Note : wherever applicable write java syntax **)**
Applet Life cycle –

Java Code :

Output: ***(note: add screen shots)***

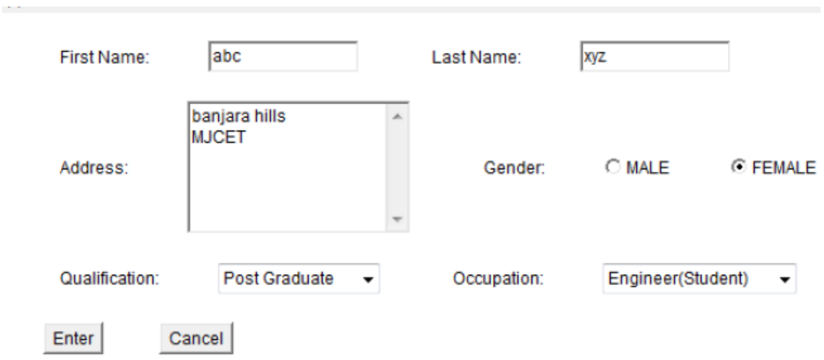
Conclusion :
I have studied ……
Exceptions handled …….

Experiment Number : 09

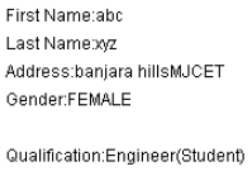
Aim : To study Swing components

Problem Statement:

a) Write a program to design login form and check valid user
b) Write a program to design calculator and perform simple arithmetic operation
c) write a program to design student college form (Refer image 1)
d) Write a program for java online test (Minimum 3 questions). Display score
e) Write a program to handle mouse listener
f) Write a program to create File, Edit menu. File menu have new, open, close and edit menu with cut, copy, paste options



First Name: abc
Last Name: xyz
Address: banjara hills
MJCET
Gender: ○ MALE ● FEMALE
Qualification: Post Graduate
Occupation: Engineer(Student)



```
First Name:abc
Last Name:xyz
Address:banjara hillsMJCET
Gender:FEMALE

Qualification:Engineer(Student)
```

Outcome : Students will able to apply swing components

Theory :

(Note : wherever applicable write java syntax)

Describe JComponents which you will use in program –

Java Code :

Output: ***(note: add screen shots)***

Conclusion :

I have studied to design GUI using JComponents such as ……

I have handled events …….. using ……….listeners

	G.S. MANDAL’S **MAHARASHTRA INSTITUTE OF TECHNOLOGY, AURABGABAD**	
	LAB WORK INSTRUCTION SHEET	
	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING	
CLASS: T.Y. B. TECH PART: 1 (2019-20)	**LAB:**	**SUBJECT: CSE303-Programming in Java**

Experiment Number : 10
Aim : To study java database connectivity
Problem Statement:
Mysql Database : Organization , Table : Dept(id int, dname varchar(02), lab int)

a) Write a program to insert ,delete, update and select records using statement
b) Write a program to insert ,delete, update and select records using PreparedStatement Take input from user.

Outcome : Students will able to connect mysql and java
Theory :
(Note : wherever applicable write java syntax)

Write steps to connect database ?

Describe following with methods :
DriverManager, Connection,Statement,PreparedStatement,ResultSet

Java Code :
Output: ***(note: add screen shots)***

Conclusion :