Laboratory Manual

For

Computer Workshop-II

(OOPM Using Java)

Lab

(UCSPC408)

Semester-IV

Computer Engineering

D. Y. Patil University Pune, Ambi

Certificate

This is to certify that Mr. /Miss……

Roll No……. and Examination seat No……..

of Class …… Branch ……. has completed all the Practical satisfactorily in Subject **Workshop-II (OOPM Using Java) (UCSPC408)** during the semester ……. For the academic year 20…….. to 20…….. as prescribed in the curriculum.

Place:…….....

Date: .....

Course Teacher **Head of the Department** **Dean**

List of Practical and Assessment

Sr. no	Title of Experiment	Page No.	Date of performan	Date of submission	Assessment marks (10)	Dated sign.
1	Programs on Basic programming constructs like branching and looping					
2	Program on accepting input through keyboard.					
3	Programs on class and objects					
4	Program on method and constructor overloading.					
5	Program on Packages					
6	Program on 2D array, strings functions					
7	Program on String Buffer and Vectors					
8	Program on types of inheritance					
9	Program on Multiple Inheritance					
10	Program on abstract class and abstract methods.					
11	Program using super and final keyword					
12	Program on Exception handling					
13	Program on user defined exception					
14	Program on Multithreading					

Instructions for Students

Student shall read the points given below for understanding the theoretical concepts and practical applications.-

1. Students shall listen carefully the lecture given by teacher about importance of subject, learning structure, course outcomes.
2. Students shall organize the work in the group of two or three members and make a record of all observations.
3. Students shall understand the purpose of experiment and its practical implementation.
4. Students shall write the answers of the questions during practical.
5. Student should feel free to discuss any difficulty faced during the conduct of practical.
6. Students shall develop maintenance skills as expected by the industries.
7. Students shall attempt to develop related hands on skills and gain confidence.
8. Students shall refer technical magazines, websites related to the scope of the subjects and update their knowledge and skills.
9. Students shall develop self-learning techniques.
10. Students should develop habit to submit the write-ups on the scheduled dates and time.

Practical No. 1

Title: Develop a Programs on Basic programming constructs like branching and looping

I. Practical Significance

Java Branching Statements are used to transfer control from one point to another in the code. Branching Statements are defined by the three keywords, break, continue, and return.

II. Minimum Theoretical Background

Java Branching Statements:

1. break

The break statement is used for terminating a loop based on a certain condition.

Example: stops the loop when i is equal to 4:

```
for (int i = 0; i < 10; i++)
{
 if (i == 4)
{
 break;
 }
 System.out.println(i);
}
```

2. continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

```
for (int i = 0; i < 10; i++)
{
 if (i == 4)
```

```
{
 continue;
 }
 System.out.println(i);
}
```

3. return

The return statement is also a branching statement, which allows us to explicitly return value from a method. The return statement exits us from the calling method and passes the control flow to where the calling method is invoked. Just like the break statement.

Syntax:

```
return value;
```

Java Looping Statements:

Loops can execute a block of code as long as a specified condition is reached.

1. while

2. do…while

3. for

1. while

The while loop loops through a block of code as long as a specified condition is true:

Syntax:

```
while (condition)
{
 // code block to be executed
}
```

In the example below, the code in the loop will run, over and over again, as long as a variable (i) is less than 5:

```
int i = 0;
while (i < 5)
```

```
{
System.out.println(i);
i++;
}
```

2. Do…while

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

```
do {
 // code block to be executed
}
while (condition);
```

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example

```
int i = 0;
do {
 System.out.println(i);
 i++;
}
while (i < 5);
```

3. for Loop

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

Syntax :

```
for (statement 1; statement 2; statement 3)
 {
```

```
// code block to be executed
}
```

The example below will print the numbers 0 to 4:

Example:

```
for (int i = 0; i < 5; i++)
{
 System.out.println(i);
}
```

4. Nested Loops

It is also possible to place a loop inside another loop. This is called a nested loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Example:

```
// Outer loop
for (int i = 1; i <= 2; i++)
 {
 System.out.println("Outer: " + i); // Executes 2 times
 // Inner loop
 for (int j = 1; j <= 3; j++)
{
 System.out.println(" Inner: " + j); // Executes 6 times (2 * 3)
 }
}
```

5. for-Each Loop

There is also a "for-each" loop, which is used exclusively to loop through elements in an array:

Syntax :

```
for (type variableName : arrayName)
{
 // code block to be executed
```

```
}
```

The following example outputs all elements in the cars array, using a "for-each" loop:

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars)
 {
 System.out.println(i);
}
```

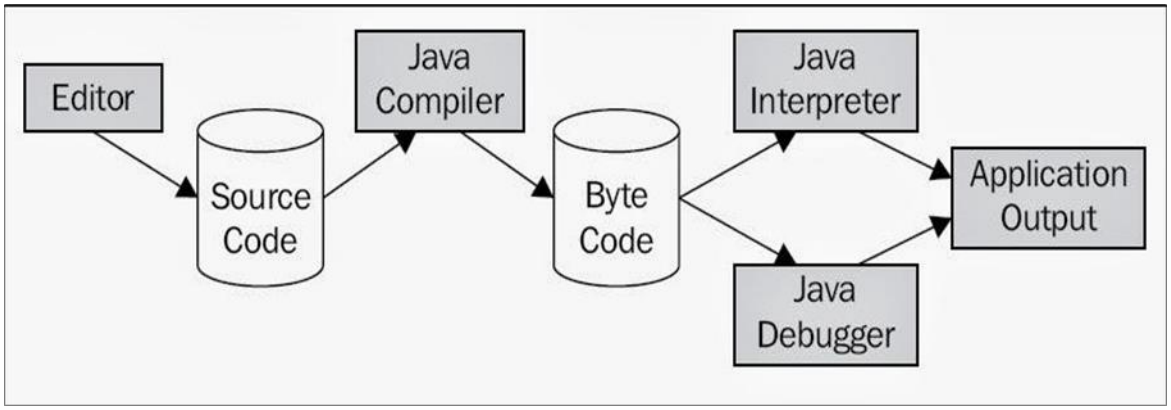


Fig. 1 Java Program Execution

1. Procedure Installation for Java Software:

(On a PC loaded with windows OS - 2000/2003/2007 onwards with notepad)

1. Download JDK (jdk 1.4.0 onwards)

Visit the https://www.oracle.com/technetwork/java/javase/downloads/index.html Download the windows version to suitable folder.

2. Double click the setup file.

3. Follow onscreen instruction.

4. When the setup is done, the complete screen appears, click on the 'Finish' Button. This completes the installation of JDK. To ensure the JDK installation / to determine the java version, type the following command at the MS Dos prompt:

<system prompt> java -version

It should show the output similar to following:

Fig. 2 Java Version

If not then set 'path' environment variable

1. Go to start -> control panel -> system

2. System properties dialogbox will appear.

3. Select 'advanced tab' -> environment variables.

4. In the system variable list, select path and click 'edit'

5. Edit the system variable dialogbox appears. In the variable value field, append the path to the JDK bin directory (generally 'c:\program files\java\jdkl.6.0\bin' at the end. Use semicolon to separate the path of bin directory from the rest of values already available. Click 'Ok'.

6. Similarly set 'classpath' environment variable

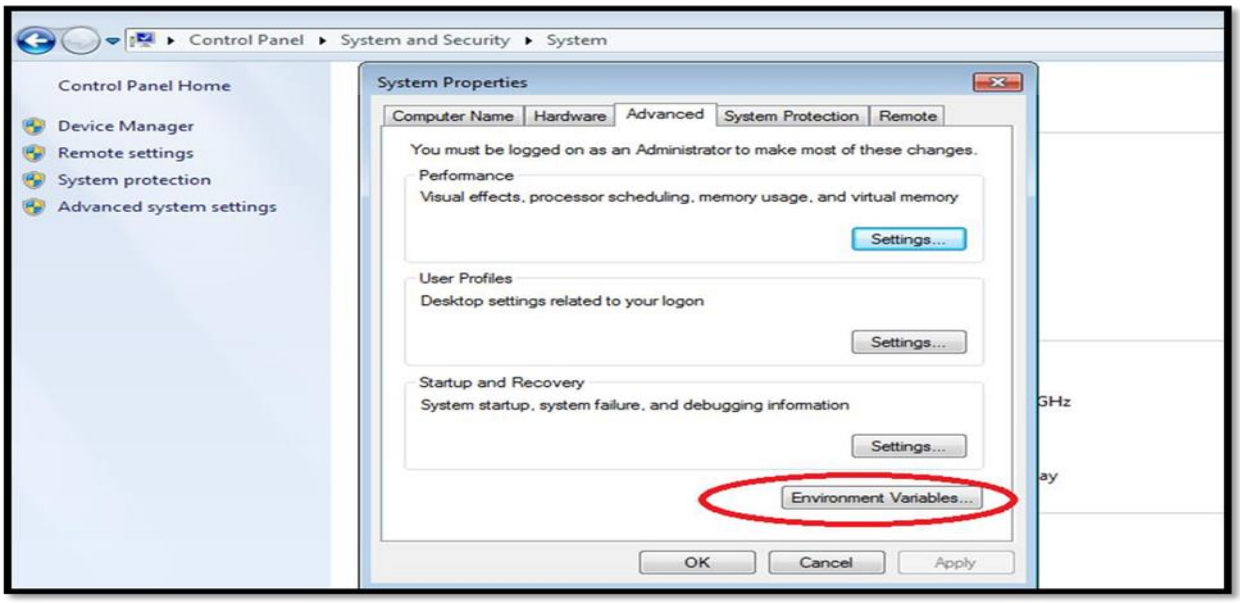


Fig. 3 Environment variable

Note: Follow the similar instructions for other platforms (say Unix, Linux,Mac) with appropriate jdk download.

1. Using an Eclipse for Java

Eclipse (@ www.eclipse.org) is an open-source Integrated Development Environment (IDE) supported by IBM. Eclipse is popular for Java application development (Java SE and Java EE) and Android apps.

Installing Eclipse 4.7.2 (Oxygen 2) for Java Developers

To use Eclipse for Java programming, you need to first install Java Development Kit (JDK).

1. Download Eclipse from https://www.eclipse.org/downloads.

Under "Get Eclipse Oxygen" Click "Download Packages".

2. To install Eclipse, unzip the downloaded file into a directory (e.g.,"d:\myproject").

2. Testing setup using small program:

Steps for editing and executing java program:

Using an editor (e.g. Notepad)

1. Open notepad

2. Write the program (calledjava source code) in notepad

3. Save the file as 'filename.java' in some directory. The filename must be same as the classname containing main() method.

3. Open MS-Dos prompt.

1. Change the directory containing to the one containing the program.

2. Compile the program by using the command javac<filename.java>

3. Execute/ Run the program by using the command java <filename>

4. Using Eclipse

1. Launch Eclipse by running "eclipse.exe" from the Eclipse installed directory.

2. Choose an appropriate directory for your workspace

3. To create a new Java project using "File" menu "New" "Java project"

4. In "JRE", select "Use default JRE (currently 'JDK9.0.x')". But make sure that your JDK is 1.8 and above.

5. In "Project Layout" menu, select "Use project folder as root for sources and class files".

6. Push "Finish" button.

7. In the "Package Explorer" (left pane) Right-click on "FirstProject New Class.

8. Write a program

9. Compile and execute program.

10. Observe output on the console panel.

Sample program:

```
Class HelloWorld
{
public static void main(String args[ ] )
```

```
{
System.out.println("Welcome to Hello World program");
}
}
```

5. Output:

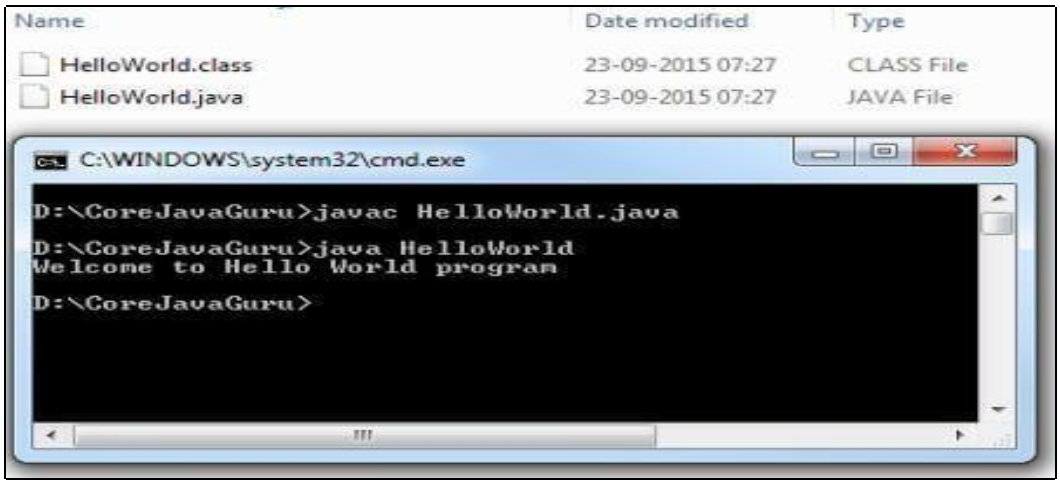


Fig 4. Output of the Program

III. Practical Related Questions

1. List operators used in looping statement.

2. Write down program using branching statements.

3. Write down program for each Looping statements.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 2

Title: Program on accepting input through keyboard.

I. Practical Significance:

The Scanner class is used to get user input, and it is found in the java.util package.To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings and nextInt() method is used to read integer.

II. Minimum Theoretical Background

Java User Input:

1. Scanner class:

The Scanner class is used to get user input, and it is found in the java.util package. To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation.

• **nextLine() method:** which is used to read Strings and

• **nextInt() method:** which is used to read integer.

```
import java.util.Scanner; // Import the Scanner class
class Main
{
 public static void main(String[] args)
{
 Scanner myObj = new Scanner(System.in); // Create a Scanner object
 System.out.println("Enter username");
 System.out.println("Enter user Number");

 String userName = myObj.nextLine(); // Read user input
 int userNo = myObj.nextInt(); // Read user input
```

```
System.out.println("Username is: " + userName); // Output user input
System.out.println("Username is: " + userNo); // Output user input
 }
}
```

III. Practical Related Questions

1. What is Packages?

2. Write Program on accept student data from user and print it?

3. What is user defined package explain with example?

(Space for Answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 3

Title: Develop a Programs on class and objects.

I. Practical Significance:

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods.

II. Minimum Theoretical Background

What is class?

Class is a group of variables of different data types and a group of methods. It is a user-defined blueprint or prototype from which objects are created.

Syntax:

```
access_modifier class <class_name>
{
 data member;
 method;
 constructor;
 nested class;
 interface;
}
```

Example:

```
public class Main
{
 int x = 5;
}
```

What is object?

An object in Java is a basic unit of Object-Oriented Programming and represents real-life entities. Objects are the instances of a class that are created to use the attributes and methods of a class.

Syntax:

classname onjectname = new classname ();

Example:

Main myObj = new Main();

III. Practical Related Questions

1. What is class explain with example?

2. What is object explain with example?

3. Write down simple class and object program with two methods.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 4

Title: Develop a Program on method and constructor overloading.

I. Practical Significance:

Constructor in Java or any programming language is a special type of method that is used to initialize the object. Different types of constructors are used to initialize the object in different way. The student will be able to use different types of constructor for creating the object.

II. Minimum Theoretical Background

Types of constructors

1. Default constructor (No-argument constructor)-

Syntax :

```
class_name()
{
Statements for initialize the data members.
}
```

2. Parameterized constructor General

Syntax:

```
class_name(parameter list)
 {
Statements for initialize the data members.
}
```

3. Copy Constructor

Syntax:

```
class_name (classname reference)
{
```

```
Statements for initialize the data members.
}
```

III. Practical Related Questions

1. Does constructor return a value?

2. Specify the situation when the default constructor is provided by the system.

3. How constructor overloading can be done?

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 5

Title: Develop a Program on Packages.

I. Practical Significance:

Packages are collection of classes and interfaces. Importing of Packages which helps in Reusability. Better organization of the classes and interfaces helps in resolving the name conflicts.

II. Minimum Theoretical Background

What is Package

It is a collection of similar types of classes, interfaces and sub-packages.

Packages are categorized in two form, built-in package and user-defined package

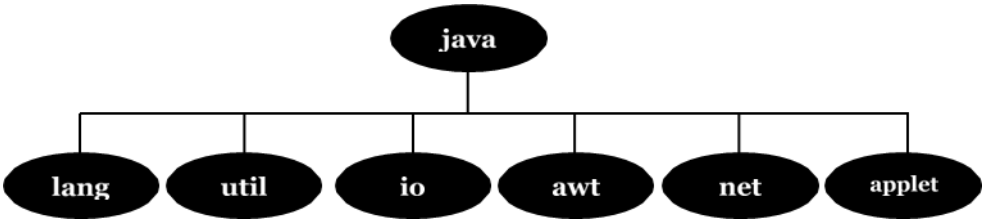


Fig. 7 Frequently Used API Packages

Using System Packages

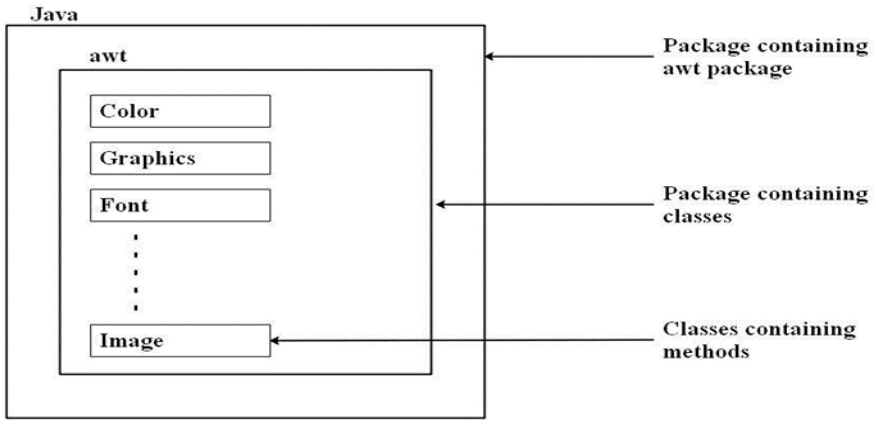


Fig. Hierarchical representation of java.awt package

Accessing the classes stored in a package:

Method 1: Using import Statements:

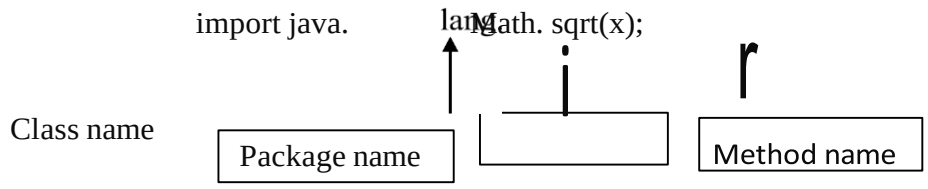
1. The first method is to use the **Fully qualified class name** of the class:

import pacakgename. Classname;

Example:

import java.lang.Math;

This statement imports the class Math and therefore class name can be used directly. It is not necessary to use the package name to qualify the class.



Method 2: Shortcut Approach

import packagename. *;

import java.lang.*;

This statement imports **every class** contained in the specified package. Above statement will bring **all the classes** of java.lang package.

```
Creating Packages
package PackageName; //package declaration
public class FirstClass // class definition
{
// (body of class)
}
```

Package Hierarchy

```
Package firstPackage.secondPackage;
```

Accessing a Package

A java system package can be accessed either using a fully qualified class name or using a shortcut approach through the import statement.

```
Syntax:
import package! [. package2][.package3].classname;
```

package1 is the name of the outer package, package2 is the name of the package that is inside the packagel.

Package hierarchy consists of any number of packages. Finally the explicit classname is specified.

```
Example:
import firstPackage.secondPackage.MyClass; // fully qualified class name
OR
import packagename. *;
import firstPackage. *; //shortcut approach
```

III. Practical Related Questions

1. Name some of java packages.

2. Can we import same class/package twice? Will the JVM load the package twice at run time?

3. Write fully qualified and shortcut class naming approach with examples.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 6

Title: Develop a Program on 2D array, strings functions.

I. Practical Significance:

Arrays store homogeneous data i.e same type of data in consecutive memory locations which will help to fetch data in constant access time. Students will be able to use array to refer multiple values with same name. The String class has a set of built-in methods that you can use on strings.

II. Minimum Theoretical Background

Array:

1. Creating an array:

```
dataType[] arrayRefVar; or
dataType arrayRefVar[]; arrayRefVar = new type [size];
```

2. Array of Objects:

```
Class_name array_name = new class_name[size];
```

3. Types of arrays:

1. One Dimensional Example:

```
int [] intArray = new int[20];
```

2. Multi Dimensional Example:

```
int[][] intArray = new int[l0][20]; //a 2D array or matrix
int[][][] intArray = new int[l0][20][10]; //a 3D array
```

String:

String is basically an object that represents sequence of char values. An array of characters works same as Java string.

```
Example:
String s="javatpoint";
```

III. Practical Related Questions

1. What is use of new operator in defining an array?

2. In 2D array which dimension is optional at the declaration of array?

3. Explain String functions?

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 7

Title: Develop a Program on String Buffer and Vectors.

I. Practical Significance:

Vector implements a dynamic array. Vector hold different number of objects. Students will be able to use Vectors in the program efficiently. String Buffer is a class in Java that represents a mutable sequence of characters. It provides an alternative to the immutable String class, allowing you to modify the contents of a string without creating a new object every time.

II. Minimum Theoretical Background

Vector class methods:

Sr. No.	Syntax	Task Performed
1.	void addElement(Object ob)	Adds the specified component to the end of this vector increasing its size by one
2.	int capacity()	Returns the current capacity of this vector
3.	boolean contains(Object elem)	Tests if the specified object is a component in this vector.
4.	void clear()	Removes all the elements from this vector
5.	Object elementAt(int index)	Returns the component at specified index
6.	Enumeration elements()	Returns enumeration of components of this vector
7.	Object firstElement()	Returns first component of this vector
8.	Object lastElement()	Returns last component of this vector
9.	int indexOf(Object elem)	Searches for the first occurrence of given argument.
10.	void insertElementAt(Object obj, int index)	Inserts specified object as a component at the specified index position.
11.	void removeElementAt(int index)	Removes the element at specified position in the vector
12.	Boolean removeElement(Object obj)	Removes first occurrence of the argument from the vector

13.	int size()	Returns number of components in the vector
14.	void copylnto(Object[] array)	Copies the components of vector into specified array.

String buffer:

```
public class StringBufferExample
{
 public static void main(String[] args)
 {
 StringBuffer sb = new StringBuffer();
 sb.append("Hello");
 sb.append(" ");
 sb.append("world");
 String message = sb.toString();
 System.out.println(message);
 }
}
```

III. Practical Related Questions

1. State difference between size() and capacity() method of Vector class.
2. Differentiate between addElement() and insertElementAt() methods of Vector class.
3. Explain String Buffer.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 8

Title: Develop a Program on types of inheritance.

I. Practical Significance:

Inheritance helps to reuse the existing class properties to derive a new class with additional properties. It helps to reduce the memory space, time, frustration and increases the reliability of code. Different types of Inheritance are used to extend the classes in different ways. The student will be able to use different types of inheritance.

II. Minimum Theoretical Background Inheritance:

The process of deriving a new class from an old class is called as Inheritance. Types of **Inheritance:**

1. Single Inheritance

2. Multiple Inheritance

3. Hierarchical Inheritance

4. Multilevel Inheritance

Java does not directly implement multiple inheritance, it is implemented using a secondary inheritance path in the form of interfaces.

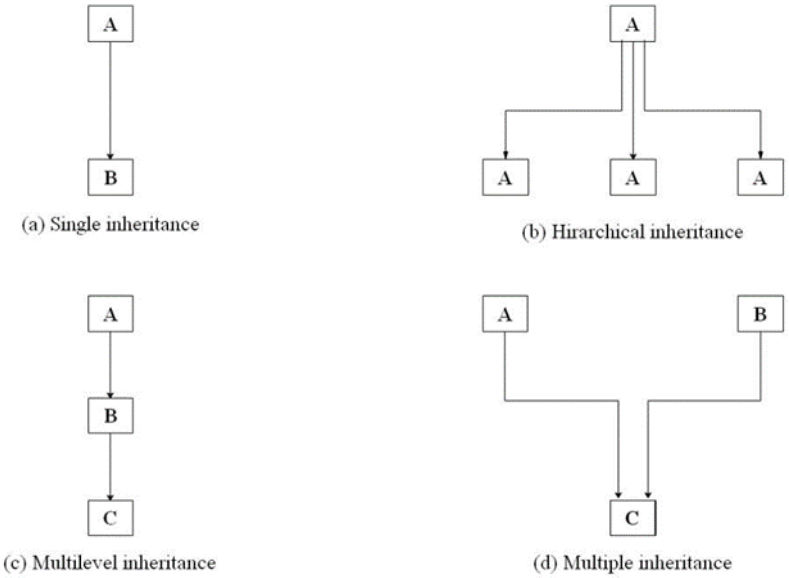


Fig. 6 Forms of Inheritance

Defining a subclass:

```
Class subclassname extends superclassname
{
variables declaration; methods declaration;
}
```

The keyword extends indicates that the properties of the superclassname are extended to the subclassname.

III. Practical Related Questions

1. Justify: Java does not support multiple inheritance.
2. Specify the conditions when the super keyword can be used.
3. Write the importance of final variables and methods.
4. Specify the conditions which needs to be satisfied while using the abstract classes.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 9

Title: Develop a Program on Multiple Inheritance.

I. Practical Significance:

Deriving a new class from multiple super/parent classes is known as multiple inheritance. Java doesn't allow multiple inheritance to avoid the ambiguity caused by the overriding methods of the classes. Interface is used to achieve multiple inheritance mJava.

II. Minimum Theoretical Background

1. Interfaces

Java does not support multiple inheritance. Classes in java cannot have more than one superclass. A large number of real-time applications require use of multiple inheritance where no. of methods are inherited from several classes. Multiple inheritance proves difficult and adds complexity to the language. Java provides an alternative approach called as interfaces to support to the concept of multiple inheritance.

Defining Interfaces

An interface is like a class. Interfaces also contain methods and variables but with a major difference. The difference is that interfaces define only abstract methods and final fields.

Interfaces do not specify any code to implement these methods and data fields contain only constants.

Syntax:

```
interface InterfaceName
{
variables declaration; methods declaration;
}
```

Where interface is the keyword and InterfaceName is any valid java variable. Variables are declared as:

```
static final type VariableName=Value;
```

Example:

```
return_type methodName1(parameter_list);
```

Interface Definition:

```
Interface Item
{
static final int id=100;
static final String name="ABC"; void display ();
}
```

Extending Interfaces

Interfaces can be extended. The new subinterface will inherit all the members of the superinterface.

```
interface name2 extends name1
{
body of name2
}
```

```
 Example:
interface ItemConstants
{
int id=l00;
string name="ABC";

//All constants in one interface
 }
interface Item extends ItemConstants
{
void display( );
}
All methods in other interface
```

2. Multiple interfaces:

```
interface ItemConstants
{
int id=l00;
string name="ABC";
}
interface ItemMethods
{
void display( );
}
interface Item extends ltemConstants, ItemMethods
{
}
```

3. Implementing Interfaces:

Interfaces are used like superclasses whose features/properties are inherited by classes.

It is mandatory to define a class that inherits the given interface.

```
class classname implements interfacename
{
body of classname
}
```

The class classname "implements" the interface interfacename.

```
class classname extends superclass implements interface1, interface2,....
{
body of classname
}
```

III. Practical Related Questions

1. Differentiate between Class and Interface.

2. Write similarities between interfaces and classes.

3. Write advantages of interfaces.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 10

Title: Develop a Program on abstract class and abstract methods.

I. Practical Significance:

Data abstraction is the process of hiding certain details and showing only essential information to the user.

II. Minimum Theoretical Background

Abstract class:

It is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Abstract methods:

Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

Example:

```
abstract class Bike{
 abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

III. Practical Related Questions

1. Explain abstract class with example.

3. Explain abstract method with example.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 11

Title: Develop a Program using super and final keyword.

I. Practical Significance:

super is used to refer to the parent class, whereas final is used to create constants, make variables immutable, and prevent method overriding or class inheritance.

II. Minimum Theoretical Background

1. super keyword:

The super keyword is used to refer to the immediate parent class object. It can be used to call the constructor, methods, or variables of the parent class. This is particularly useful when there is a method or variable in the child class with the same name as that in the parent class, and you want to differentiate between the two.

Example of using the super keyword:

```
class Parent {
 String message = "Hello from Parent";

 void display() {
 System.out.println(message);
 }
}

class Child extends Parent {
 String message = "Hello from Child";

 void display() {
 super.display(); // Calling the display() method of the parent class
 System.out.println(message);
 }
}
```

```
public class Main {
 public static void main(String[] args) {
 Child obj = new Child();
 obj.display();
 }
}
```

Output:

```
Hello from Parent
Hello from Child
```

2. final keyword:

The final keyword in Java is used to declare constants, make variables immutable, and prevent method overriding or class inheritance. When a variable is declared as final, its value cannot be changed once initialized. When a method is declared as final, it cannot be overridden by subclasses. When a class is declared as final, it cannot be subclassed.

Examples of using the final keyword:

```
// Example of final variable
class Example {
 final int value = 10; // Once initialized, 'value' cannot be changed
}

// Example of final method
class Parent {
 final void display() {
 System.out.println("This method is final and cannot be overridden.");
 }
}

class Child extends Parent {
 // Error: display() in Child cannot override display() in Parent
 // Attempting to override a final method
 void display() {
```

```
 System.out.println("This method cannot override the final method in the parent class.");
 }
}

// Example of final class
final class FinalClass {
 // Class body
}
// Error: Cannot inherit from final FinalClass
// class AnotherClass extends FinalClass {}
```

III. Practical Related Questions

1. Explain super keyword with syntax and example.

2. . Explain final keyword with syntax and example.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 12

Title: Develop a Program on Exception handling.

I. Practical Significance:

Exception handling is crucial for writing robust and reliable Java applications, as it helps in identifying and handling errors gracefully, ensuring smoother program execution.

II. Minimum Theoretical Background

Exception handling:

Exception handling in Java is a mechanism to deal with runtime errors or exceptional situations that occur during the execution of a program. These exceptions can be caused by various factors such as invalid user input, network issues, file I/O errors, etc. Exception handling helps in gracefully handling these situations, preventing the program from crashing and allowing for appropriate actions to be taken.

Here's a basic example of exception handling in Java:

```
import java.util.Scanner;
public class Main {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 System.out.print("Enter a number: ");

 try {
 // Attempt to read an integer from user input
 int num = scanner.nextInt();

 // Perform division by zero to provoke an ArithmeticException
 int result = 10 / num;

 // This line will not be executed if an exception occurs
 System.out.println("Result: " + result);
 } catch (ArithmeticException e) {
```

```
// Catch and handle ArithmeticException
System.out.println("Error: Division by zero is not allowed.");
} catch (Exception e) {
// Catch any other exceptions
System.out.println("An error occurred: " + e.getMessage());
} finally {
// Finally block always executes, regardless of whether an exception occurred
scanner.close(); // Close the scanner to avoid resource leak
}

System.out.println("Program continues after exception handling.");
}
}
```

In this example:

- We use a **try** block to enclose the code that might throw an exception.
- If an exception occurs during the execution of the **try** block, the corresponding **catch** block is executed.
- Multiple **catch** blocks can be used to handle different types of exceptions.
- The **finally** block is optional and always executes, whether an exception occurs or not. It's typically used to perform cleanup tasks like closing resources.
- If no exceptions occur, the code in the **try** block is executed sequentially, and the **catch** and **finally** blocks are skipped.

When you run this program and provide valid input, it will execute successfully and print the result. However, if you provide zero as input, it will throw an ArithmeticException, which will be caught by the corresponding catch block, preventing the program from crashing and displaying an error message instead.

III. Practical Related Questions

1. What is try and catch explain with example?

2. Write down program on Excepting handling.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	

Practical No. 13

Title: Develop a Program on user defined exception.

I. Practical Significance:

User-defined exceptions allow you to create more meaningful and specific exception types for your application, improving readability and maintainability by providing clear indications of what went wrong.

II. Minimum Theoretical Background

1. user defined exception:

In Java, you can define your own custom exceptions by extending the Exception class or one of its subclasses. This allows you to create exceptions that are specific to your application's domain or requirements.

Example of how to create a user-defined exception:

```
// Custom exception class extending Exception
class CustomException extends Exception {
 // Constructor with a custom error message
 public CustomException(String message) {
 super(message);
 }
}

// Class that demonstrates the use of the custom exception
public class Main {
 // Method that throws the custom exception
 public static void validate(int age) throws CustomException {
 if (age < 18) {
 throw new CustomException("Age must be 18 or above."); // Throwing the custom exception
 } else {
 System.out.println("Valid age.");
 }
 }
```

```
public static void main(String[] args) {
int age = 15;

try {
// Calling the method that may throw the custom exception
validate(age);
} catch (CustomException e) {
// Catching and handling the custom exception
System.out.println("Exception caught: " + e.getMessage());
}
}
}
```

In this example:

- We define a custom exception class **CustomException** that extends the **Exception** class.
- We provide a constructor for the **CustomException** class that takes a custom error message as a parameter and passes it to the superclass constructor using **super(message)**.
- We define a method **validate()** that checks if the given age is less than 18 and throws a **CustomException** if it is.
- In the main() method, we call the validate() method within a try-catch block to catch the CustomException if it's thrown.

When you run this program with an age less than 18, it will throw the CustomException, which will be caught by the catch block, and the error message "Age must be 18 or above." will be displayed. If the age is 18 or above, the program will execute without throwing any exceptions.

User-defined exceptions allow you to create more meaningful and specific exception types for your application, improving readability and maintainability by providing clear indications of what went wrong.

III. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Is it possible to start a thread twice?

2. Can we call the run() method instead of start()?

3. Differentiate between notify() and notifyAll( )?

4. Explain the use of keyword synchronized.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				Dated signature of Teacher
Process Related(4)	Product Related(4)	Attendance(2)	Total (10)	

Practical No. 14

Title: Develop a Program on Multithreading.

I. Practical Significance:

Multithreading technique in java helps to run multiple programs or a processes concurrently by utilizing the maximum CPU time. Multithreaded technique is implemented by creating, declaring, extending, implementing by thread. Student will be able to implement different types of thread methods by assigning the priority to illustrate simultaneous execution of thread operation.

II. Minimum Theoretical Background

1. Multithreading

Multithreading in Java allows multiple tasks to execute concurrently within a single Java program. Each thread represents an independent flow of execution, allowing different parts of your program to run simultaneously. Multithreading is commonly used to perform tasks concurrently, improve performance, and utilize available resources efficiently.

Example demonstrating multithreading in Java:

```
// Define a class that extends Thread
class MyThread extends Thread {
 // Override the run() method to define the task for the thread
 public void run() {
 for (int i = 0; i < 5; i++) {
 System.out.println("Thread: " + Thread.currentThread().getId() + " - Count: " + i);
 try {
 Thread.sleep(1000); // Pause for 1 second
 } catch (InterruptedException e) {
 System.out.println("Thread interrupted.");
 }
 }
 }
}
```

```
public class Main {
 public static void main(String[] args) {
 // Create multiple threads of MyThread class
 MyThread thread1 = new MyThread();
 MyThread thread2 = new MyThread();

 // Start the threads
 thread1.start();
 thread2.start();
 }
}
```

In this example:

- We define a class MyThread that extends the Thread class.
- Inside MyThread, we override the run() method, which contains the code to be executed by the thread.
- In the run() method, a loop prints numbers from 0 to 4 along with the ID of the thread. It then pauses for 1 second between each iteration using Thread.sleep(1000) to simulate some work being done.
- In the main() method, we create two instances of MyThread, representing two separate threads.
- We start each thread using the start() method, which initiates the execution of the run() method in each thread.

When you run this program, you'll see both threads executing concurrently, printing their respective counts at approximately one-second intervals. The actual output may vary depending on the scheduler and system load, but you'll observe interleaved output from both threads.

III. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Is it possible to start a thread twice?

2. Can we call the run() method instead of start()?

3. Differentiate between notify() and notifyAll( )?

4. Explain the use of keyword synchronized.

(Space for answer)

IV. Assessment Scheme

Marks Obtained				**Dated signature of Teacher**
Process Related(4)	**Product Related(4)**	**Attendance(2)**	**Total (10)**	