

M.Sc. - II
Computer Applications
(SEMESTER - II)

CSA4211
Computer Applications Practical – III
(Lab Based on Data Mining and Core Java)

Work-book(20_-20_)

Name : _____

Roll No. : _____

Core Java

Assignment Completion Sheet

Assignment No.	Assignment Name	Marks	Viva
I	Constructor and Static Function / Variable		
II	Packages		
III	Abstract Class and Inheritance		
IV	Interface and Inheritance		
V	File Handling		
VI	Applets		
VII	AWT Interface		
VIII	Swing Interface		
IX	Event Listeners		
X	Applet / Swing Interface / Adapter Class		
Total		/ 50	/ 20
		/ 10	

This is to certify that Mr. / Ms. _____
has completed ____ number of experiments and has obtained ____ marks out of 10 and also
attempted ____ viva-voce and has obtained _____ marks out of 20.

Name and Sign of batch In-charge

**Head,
Department of Computer Science**

Date:

Examiner1

Examiner2

Constructors and Static function / variable

Ready Reference

File naming convention for java programs

- Any java program must be saved in a file with extension “.java”.
- A java file can contain at most one public class.
- If a java file contains a public class then the name of the file must be same as the name of the public class
- If a java file contains no public class then the name of the file must be same as any of the non-public class present in that file.
- Smallest non-empty java file, which is compilable, must have at least one class declaration.

Class

For any variable x in a programming language, the type of x defines what kinds of operations are valid on that variable. Java, like every programming language, has a set of primitive types to represent integer, character, real etc. Java also allows user to create new types by means of class declaration. A class is an encapsulation of data and the methods operating on this data in a single unit of type. A class in Java is written as below.

```
class intro
{

}
}
```

A class in Java is called a public class if the keyword public is added before the keyword Class in a class definition. For example the code creates a public class newintro

```
public class newintro
{

}
}
```

Object

Object is an instantiation of a class type. For the example

```
intro var;
```

where var is any string.

Above syntax declares that var is a reference/handle of type intro. It is to be noted that the above syntax only creates a handle/reference to an intro object. Actual object is not created by this syntax. . Java provides a keyword new to create a new object of a given type and assign it to a reference of that type.

```
intro var = new intro;
```

The above syntax does two things. It first declares var to be a reference/handle of intro type. Second, it creates a new object of type intro and assign its reference/address to variable var . It is clear that many objects can be created of the same type/class . Each of these objects occupy separate memory area.

A class consists of a set of variables, also called fields, and a set of functions, also called methods. For an object of a given class, the fields and methods of that class are accessed using operator. A field of a class is of the following form [Qualifier] type variable name.

Where Qualifier is optional and can take following values,

- public: It means that this field can be accessed from outside the class by just using objectname.fieldname
- static: It means that to access this field you do not need to create an object of this class. Therefore, this field is also sometimes called as class variable/field

For the example program in listing 3, credits is a public static variable and hence can be accessed from outside this class as intro.credits . On the other hand, age is only a public variable and hence can only be accessed after creating an object of intro type as below.

```
intro var = new intro;  
var.age = 10;
```

Above code first creates an object of intro class and then assign value 10 to the field age of this object. It is to be noted that every object of a class, contains different copies of non-static variable (e.g. age) and the same copy of static variable (credits) You do not require to create an object to access the static members of a class.

```
class intro  
{  
    public int age;  
    public static int credits;  
}
```

Methods

A method/function is defined inside a class as [Qualifier] return- type function-name (arguments). A method in Java also specifies exception signature which we will discuss in detail in error handling. Two important qualifiers for methods are same as for fields,

- public: A public method can be invoked from outside the class using the object of that class.
- static: A static method can be called without creating an object of that class.

Following restrictions are applied on invocation of static methods.

- static function can not call non-static functions of that class.
- static function can not access non-static variables/fields of that class.

Overriding toString method of the Object class:

The toString method gives a string representation of an object. To over-ride the toString method for a user defined class, use the syntax:

```
public String toString()  
{  
    // return a string representation of the object  
}
```

Example

```
class Student
```

```
{  
    private int rollNumber;  
    private String name;  
  
    public String toString()  
    {  
        return "Roll Number = " + rollNumber + "Name = "+name;  
    }  
}
```

Static fields and methods:

Static fields are class variables which have the “static” modifier. They don’t belong to the instance but belong to the class. Static methods are used to access static members of a class.

To access a static member, use the following syntax:

```
ClassName.staticMember
```

```
ClassName.staticMethod(arguments)
```

Declaring an array of references:

```
ClassName[] arrayName = new ClassName[size];
```

Creating an array of objects:

For each reference in the array

```
{  
    Create an object using new  
}
```

Example:

```
Student[] studentArray = new Student[10];  
for(i=0; i<10; i++)  
    studentArray[i] = new Student();
```

Command line arguments:

We can pass information to main from the command line using command line arguments. These are stored in an array of Strings which is passed to main as an argument.

```
public static void main(String[] args)
```

Here, args is the name of the array. The total number of arguments can be obtained using args.length . To access each argument, use a for loop as shown:

```
for(int i=0; i<args.length ; i++)
```

```
System.out.println("Argument "+ i + " = " + args[i]);
```

To convert the argument from String to any type, use Wrapper classes.

Method	Purpose
Byte.parseByte	Returns byte equivalent of a String
Short.parseShort	Returns the short equivalent of a String
Integer.parseInt	Returns the int equivalent of a String
Long.parseLong	Returns the long equivalent of a String
Float.parseFloat	Returns the float equivalent of a String
Double.parseDouble	Returns the double equivalent of a String

Simple I/O :

To read a String from the console, use the following code:

```
InputStreamReader isr = new InputStreamReader(System.in);
```

```
BufferedReader br = new BufferedReader(isr);
```

Or

```
BufferedReader br = new BufferedReader(new
```

```
InputStreamReader(System.in));
```

For this, you will have to write the following statement at the beginning:

```
import java.io.*;
```

Constructor

Every class in Java has a public method by default, it is called Constructor of that class . The name of the constructor function is same as the name of the class without any return type, but it can take a set of arguments like any method. This method is called when an object of this class is created using new operator as shown earlier.

```

class intro
{
    public int age;
    public static int credits;
    public intro(int a)
    {
        age = a;
    }
    public void setage(int newage)
    {
        age = newage;
    }
    public static int getcredit ()
    {
        return credits;
    }
}

```

In the example, static function getcredit returns the value of static field credits. Function setage takes an integer argument and set the value of the field age to this value. Return type of setage is void as it does not return any value, while the return value of getcredit is int . We also have a constructor in this class which takes an integer and assign it to field Age of that object. Constructors are mainly used to make sure that all fields of an object are initialized properly at the time of creation. Having covered the most essential aspect of a class definition in Java, now let us look at the process of compiling and executing a Java program.

Assignment – I

Constructor and Static Function / Variable

Practical Assignments

1. Define an Employee class (name, position, salary). Define a default and parameterized constructor. Override the toString method. Keep a count objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method). Also display the contents of each object.
2. Define a class MyNumber having one private int data member. Write a default constructor to initialize it to 0 and another constructor to initialize it to a value (Use this). Write methods to check whether this data member is Negative, Positive, Zero, Odd or Even. Create an object in main. Use command line arguments to pass a value to the object (Hint: convert string argument to integer) and perform the above checking.
3. Define a Student class (roll number, name, percentage). Define a default and parameterized constructor. create n objects of the Student class. Accept details from the user for each object. Define a static method “sortStudent” which sorts the array on the basis of percentage.
4. Write a program for matrix addition. The addition function should return the addition of the matrix.

Assignment Evaluation

0: Not Done 1: Incomplete 2: Late Complete

3: Needs Improvement 4: Complete 5: Well Done

Viva –Voce (0 to 2)

Signature of the Teacher

Signature of Student

Date ____/____/____

Packages

Ready Reference

Packages:

A package is a collection of related classes and interfaces. It provides a mechanism for compartmentalizing classes. The Java API is organized as a collection of several predefined packages. The java.lang package is the default package included in all java programs. The commonly used packages are:

java.lang	Language support classes such as Math, Thread, String
java.util	Utility classes such as LinkedList, Vector, Date.
java.io	Input/Output classes
java.awt	For graphical user interfaces and event handling.
javax.swing	For graphical user interfaces
java.net	For networking
java.applet	For creating applets.

Creating a package

To create a user defined package, the package statement should be written in the source code file. This statement should be written as the first line of the program. Save the program in a directory of the same name as the package.

```
package packageName;
```

Accessing a package

To access classes from a package, use the import statement.

```
import packageName.*; //imports all classes
```

```
import packageName.className; //imports specified class
```

Note that the package can have a hierarchy of subpackages. In that case, the package name should be qualified using its parent packages. Example: project.sourcecode.java

Here, the package named project contains one subpackage named sourcecode which contains a subpackage named java.

Access Rules

The access rules for members of a class are given in the table below.

Accessible to	public	protected	none	private
Same class	Yes	Yes	Yes	Yes
Class in same package	Yes	Yes	Yes	No
Subclass (in other package)	Yes	Yes	No	No
Non subclass in Other package	Yes	No	No	No

Assignment – II

Packages

Practical Assignments

1. Create a package named NumOperation having three different classes:
 - a. Prime number
 - b. Perfect Number
 - c. Armstrong number

Write a program to accept number 'n' from the user and check whether it is prime number or perfect number or Armstrong number using the above NumOperation package.

2. Write a program to create a Package "MScCAI" which has a class MScCAIMarks (members – SemITotal, SemIITotal). Create another package "MScCAII" which has a class MScCAIIMarks (members – SemITotal, SemIITotal). Create n objects of Student class (having rollNumber, name, MScCAIMarks and MScCAIIMarks). Add the marks of MScCAI and MScCAII. Calculate the Grade ('A' for ≥ 70 , 'B' for ≥ 60 'C' for ≥ 50 , Pass Class for ≥ 40 else 'FAIL') and display the result of the student in proper format.

Assignment Evaluation

0: Not Done	<input type="text"/>	1: Incomplete	<input type="text"/>	2: Late Complete	<input type="text"/>
3: Needs Improvement	<input type="text"/>	4: Complete	<input type="text"/>	5: Well Done	<input type="text"/>
Viva –Voce (0 to 2)	<input type="text"/>				

Signature of the Teacher

Signature of Student

Date ____/____/____

Abstract class, Inheritance and Interfaces

Ready Reference

Inheriting a class :

The syntax to create a subclass is :

```
class SubClassName extends SuperClassName
{
//class body
}
```

Example:

```
class Manager extends Employee
{ //code }
```

Types of Inheritance

1. Single
2. Multilevel
3. Hierarchical

Access in subclass

The following members can be accessed in a subclass:

- i) public or protected superclass members.
- ii) Members with no specifier if subclass is in same package.

The “super” keyword

It is used for three purposes:

- i) Invoking superclass constructor - super(arguments)
- ii) Accessing superclass members – super.member
- iii) Invoking superclass methods – super.method(arguments)

Example:

```
class A
{
    protected int num;
    A(int num) { this.num = num; }
}
class B extends A
{
    int num;
    B(int a, int b) {
        super(a); //should be the first line in the subclass constructor
        this.num = b;
    }
    void display() {
        System.out.println("In A, num = " + super.num);
        System.out.println("In B, num = " + num);
    }
}
```

Overriding methods

Redefining superclass methods in a subclass is called overriding. The signature of the subclass method should be the same as the superclass method.

```
class A
{
    void method1(int num) {
        //code
    }
}
class B extends A
{
    void method1(int x) {
        //code
    }
}
```

Dynamic binding

When over-riding is used, the method call is resolved during run-time i.e. depending on the object type, the corresponding method will be invoked.

Example:

```
A ref;
ref = new A();
ref.method1(10); //calls method of class A
ref = new B();
ref.method1(20); //calls method of class B
```

Abstract class

An abstract class is a class which cannot be instantiated. It is only used to create subclasses. A class which has abstract methods must be declared abstract. An abstract class can have data members, constructors, method definitions and method declarations.

```
abstract class ClassName
{
    ...
}
```

Abstract method

An abstract method is a method which has no definition. The definition is provided by the subclass.

```
abstract returnType method(arguments);
```

Interface

An interface is a pure abstract class i.e. it has only abstract methods and final variables. An interface can be implemented by multiple classes.

```
interface InterfaceName
```

```
{
```

```
//abstract methods
```

```
//final variables
```

```
}
```

Example:

```
interface MyInterface
```

```
{
```

```
    void method1();
```

```
    void method2();
```

```
    int size= 10; //final and static
```

```
}
```

```
class MyClass implements MyInterface {
```

```
//define method1 and method2
```

```
}
```

```
*****
```

Assignment - III

Abstract class and Inheritance

Practical Assignments

1. Define a class Employee having private members – id, name, department, salary. Define default and parameterized constructors. Create a subclass called “Manager” with private member bonus. Define methods accept and display in both the classes. Create n objects of the Manager class and display the details of the manager having the maximum total salary (salary+bonus).
2. Create an abstract class Shape with methods calc_area and calc_volume. Derive three classes Sphere(radius) , Cone(radius, height) and Cylinder(radius, height), Box(length, breadth, height) from it. Calculate area and volume of all. (Use Method overriding).
3. Define an abstract class “Staff” with members name and address. Define two sub-classes of this class – “FullTimeStaff” (department, salary) and “PartTimeStaff” (number-of-hours, rate-perhour). Define appropriate constructors. Create n objects which could be of either FullTimeStaff or PartTimeStaff class by asking the user’s choice. Display details of all “FullTimeStaff” objects and all “PartTimeStaff” objects.

Assignment Evaluation

0: Not Done	<input type="text"/>	1: Incomplete	<input type="text"/>	2: Late Complete	<input type="text"/>
3: Needs Improvement	<input type="text"/>	4: Complete	<input type="text"/>	5: Well Done	<input type="text"/>
Viva –Voce (0 to 2)	<input type="text"/>				

Signature of the Teacher

Signature of Student

Date ____/____/____

Assignment - IV

Interface and Inheritance

Practical Assignments

1. Create an interface “CreditCardInterface” with methods : viewCreditAmount(), useCard(), payCredit() and increaseLimit(). Create a class SilverCardCustomer (name, cardnumber (16 digits), creditAmount – initialized to 0, creditLimit - set to 50,000) which implements the above interface. Inherit class GoldCardCustomer from SilverCardCustomer having the same methods but creditLimit of 1,00,000. Create an object of each class and perform operations. Display appropriate messages for success or failure of transactions. (Use method overriding)
 - i. useCard() method increases the creditAmount by a specific amount upto creditLimit
 - ii. payCredit() reduces the creditAmount by a specific amount.
 - iii. increaseLimit() increases the creditLimit for GoldCardCustomers (only 3 times, not more than 5000Rs. each time)
2. Write a program to create a super class Vehicle having members Company and price. Derive 2 different classes LightMotorVehicle (members – mileage) and HeavyMotorVehicle (members – capacity-in-tons). Accept the information for n vehicles and display the information in appropriate form. While taking data, ask the user about the type of vehicle first.

Assignment Evaluation

0: Not Done	<input type="text"/>	1: Incomplete	<input type="text"/>	2: Late Complete	<input type="text"/>
3: Needs Improvement	<input type="text"/>	4: Complete	<input type="text"/>	5: Well Done	<input type="text"/>
Viva –Voce (0 to 2)	<input type="text"/>				

Signature of the Teacher

Signature of Student

Date ____/____/____

File Handling

Ready Reference

java.io.File class

This class supports a platform-independent definition of file and directory names. It also provides methods to list the files in a directory, to check the existence, readability, writeability, type, size, and modification time of files and directories, to make new directories, to rename files and directories, and to delete files and directories.

Constructors:

```
public File(String path);  
public File(String path, String name);  
public File(File dir, String name);
```

Example

```
File f1=new File("/home/java/a.txt");
```

Methods

1. boolean `canRead()`- Returns True if the file is readable.
2. boolean `canWrite()`- Returns True if the file is writeable.
3. String `getName()`- Returns the name of the File with any directory names omitted.
4. boolean `exists()`- Returns true if file exists
5. String `getAbsolutePath()`- Returns the complete filename. Otherwise, if the File is a relative file specification, it returns the relative filename appended to the current working directory.
6. String `getParent()`- Returns the directory of the File. If the File is an absolute specification.
7. String `getPath()`- Returns the full name of the file, including the directory name.
8. boolean `isDirectory()`- Returns true if File Object is a directory
9. boolean `isFile()`- Returns true if File Object is a file
10. long `lastModified()`- Returns the modification time of the file (which should be used for comparison with other file times only, and not interpreted as any particular time format).
11. long `length()`- Returns the length of the file.
12. boolean `delete()`- deletes a file or directory. Returns true after successful deletion of a file.
13. boolean `mkdir ()`- Creates a directory.
14. boolean `renameTo (File dest)`- Renames a file or directory. Returns true after successful renaming

Directories

A directory is a File that contains a list of other files & directories. When you create a File object & it is a directory, the `isDirectory()` method will return true. In this case list method can be used to extract the list of other files & directories inside.

The forms of `list()` method is

```
public String[ ] list()  
public String[ ] list(FilenameFilter filter)
```

Reading and Writing Files

In Java, all files are byte-oriented, and Java provides methods to read and write bytes from and to a file.

Two most often used stream classes are `FileInputStream` and `FileOutputStream` which create byte streams linked to files. To open a file, simply create an object of one of these classes, specifying the name of the file as an argument to the constructor. While both classes support additional, overridden constructors, the following are the forms that we will be using:

`FileInputStream(String filename)` throws `FileNotFoundException`

`FileOutputStream(String filename)` throws `FileNotFoundException`

Where `filename` specifies the name of the file to open. In both cases it throws `FileNotFoundException`.

When the use of file is over, close it by calling `close()` and defined by both `FileInputStream` and `FileOutputStream`. The syntax is: `void close()` throws `IOException`.

To read from a file, use a `read()` method, defined within `FileInputStream`.

The syntax is given as:

`int read()` throws `IOException`

Each time that it is called, it reads a single byte from the file and returns the byte as an integer value. `read()` returns -1 when the end of the file is encountered. It can throw an `IOException`.

To write to a file, use the `write()` method defined by `FileOutputStream`.

The syntax is given as:

`void write(int byteval)` throws `IOException`

this method writes the byte specified by `byteval` to the file. Although `byteval` is declared as an integer, only the low-order eight bits are written to the file. If an error occurs during writing, an `IOException` is thrown.

Assignment –V

File Handling

Practical Assignments

1. Write a program to accept a string as command line argument and check whether it is a file or directory. If it is a directory, list the contents of the directory, count how many files the directory has and display name of all files in that directory having extension .txt. If it is a file, display all information about the file (path, size, attributes etc).
2. Write a program to accept a filename as command line argument and print the number of lines, number of characters and number of words in the file.
3. Write a program to accept a file called “input.txt” as command line argument. Write all text at even line numbers from the file in “even.txt” and all text at odd line number from the file in “odd.txt”.

Assignment Evaluation

0: Not Done	<input type="text"/>	1: Incomplete	<input type="text"/>	2: Late Complete	<input type="text"/>
3: Needs Improvement	<input type="text"/>	4: Complete	<input type="text"/>	5: Well Done	<input type="text"/>
Viva –Voce (0 to 2)	<input type="text"/>				

Signature of the Teacher

Signature of Student

Date ____/____/____

Applets, AWT Interface, Swing Interface, Event Listeners and Adapter Class

Ready Reference

Graphical User Interface elements are implemented in two java packages – AWT and Swing. Swing is the newer package and swing classes are based on AWT classes.

Swing Architecture:

The design of the Swing component classes is based on the Model-View-Controller architecture, or MVC.

1. The model stores the data.
2. The view creates the visual representation from the data in the model.
3. The controller deals with user interaction and modifies the model and/or the view.

Swing Classes:

The following table lists some important Swing classes and their description.

Class	Description
Box	Container that uses a BorderLayout
JApplet	Base class for Swing applets
JButton	Selectable component that supports text/image display
JCheckBox	Selectable component that displays state to user
JCheckBoxMenuItem	Selectable component for a menu; displays state to user
JColor	Chooser For selecting colors
JComboBox	For selecting from a drop-down list of choices
JComponent	Base class for Swing components
JDesktopPane	Container for internal frames
JDialog	Base class for pop-up subwindows
JEditorPane	For editing and display of formatted content
JFileChooser	For selecting files and directories
JFormattedTextField	For editing and display of a single line of formatted text
JFrame	Base class for top-level windows
JInternalFrame	Base class for top-level internal windows
JLabel	For displaying text/images
JLayeredPane	Container that supports overlapping components
JList	For selecting from a scrollable list of choices
JMenu	Selectable component for holding menu items; supports text/image display
JMenuBar	For holding menus
JMenuItem	Selectable component that supports text/image display
JOptionPane	For creating pop-up messages
JPanel	Basic component container
JPasswordField	For editing and display of a password
JPopupMenu	For holding menu items and popping up over components
JProgressBar	For showing the progress of an operation to the user
JRadioButton	Selectable component that displays state to user; included in ButtonGroup to ensure that only one button is selected

JRadioButtonMenuItem	Selectable component for menus; displays state to user; included in ButtonGroup to ensure that only one button is selected
JRootPane	Inner container used by JFrame, JApplet, and others
JScrollBar	For control of a scrollable area
JScrollPane	To provide scrolling support to another component
JSeparator	For placing a separator line on a menu or toolbar
JSlider	For selection from a numeric range of values
JSpinner	For selection from a set of values, from a list, a numeric range, or a date range
JSplitPane	Container allowing the user to select the amount of space for each of two components
JTabbedPane	Container allowing for multiple other containers to be displayed; each container appears on a tab
JTable	For display of tabular data
JTextArea	For editing and display of single-attributed textual content
TextField	For editing and display of single-attributed textual content on a single line
JTextPane	For editing and display of multi-attributed textual content
JToggleButton	Selectable component that supports text/image display; selection triggers component to stay “in”
JToolBar	Draggable container
JToolTip	Internally used for displaying tool tips above components
JTree	For display of hierarchical data
JViewport	Container for holding a component too big for its display area
JWindow	Base class for pop-up windows

Layout Manager

The job of a layout manager is to arrange components on a container. Each container has a layout manager associated with it. To change the layout manager for a container, use the `setLayout()` method.

Syntax

```
setLayout(LayoutManager obj)
```

The predefined managers are listed below:

1. FlowLayout
2. BorderLayout
3. GridLayout
4. BoxLayout
5. CardLayout
6. GridBagLayout



Important Containers:

1. JFrame

This is a top-level container which can hold components and containers like panels.

Constructors

JFrame()

JFrame(String title)

Important Methods

Method	Description
setSize(int width, int height)	Specifies size of the frame in pixels
setLocation(int x, int y)	Specifies upper left corner
setVisible(boolean visible)	Set true to display the frame
setTitle(String title)	Sets the frame title
setDefaultCloseOperation(int mode)	Specifies the operation when frame is closed. The modes are: JFrame.EXIT_ON_CLOSE JFrame.DO_NOTHING_ON_CLOSE JFrame.HIDE_ON_CLOSE JFrame.DISPOSE_ON_CLOSE
pack()	Sets frame size to minimum size required to hold components

2. JPanel –

This is a middle-level container which can hold components and can be added to other containers like frame and panels.

Constructors

```
public javax.swing.JPanel(java.awt.LayoutManager, boolean);
```

```
public javax.swing.JPanel(java.awt.LayoutManager);
```

```
public javax.swing.JPanel(boolean);
```

```
public javax.swing.JPanel();
```

Important Components:

1. Label :

With the JLabel class, you can display unselectable text and images.

Constructors-

```
JLabel(Icon i)
JLabel(Icon I , int n)
JLabel(String s)
JLabel(String s, Icon i, int n)
JLabel(String s, int n)
JLabel()
```

The int argument specifies the horizontal alignment of the label's contents within its drawing area; defined in the SwingConstants interface (which JLabel implements): LEFT (default), CENTER, RIGHT, LEADING, or TRAILING.

Methods-

1. Set or get the text displayed by the label.

```
void setText(String)
String getText()
```

2. Set or get the image displayed by the label.

```
void setIcon (Icon)
Icon getIcon()
```

3. Set or get the image displayed by the label when it's disabled. If you don't specify a disabled image, then the look-and-feel creates one by manipulating the default image.

```
void setDisabledIcon(Icon)
Icon getDisabledIcon()
```

4. Set or get where in the label its contents should be placed. For vertical alignment: TOP, CENTER (the default), and BOTTOM.

```
void setHorizontalAlignment(int)
void setVerticalAlignment(int)
int getHorizontalAlignment()
int getVerticalAlignment()
```

2. Button

A Swing button can display both text and an image. The underlined letter in each button's text shows the mnemonic which is the keyboard alternative.

Constructors-

```
JButton(Icon I)
JButton(String s)
JButton(String s, Icon I)
```

Methods

void setDisabledIcon(Icon)

void setPressedIcon(Icon)

void setSelectedIcon(Icon)

void setRolloverIcon(Icon)

String getText()

void setText(String)

Event- ActionEvent

3. Check boxes

Class- JCheckBox

Constructors-

JCheckBox(Icon i)

JCheckBox(Icon i, boolean state)

JCheckBox(String s)

JCheckBox(String s, boolean state)

JCheckBox(String s, Icon i)

JCheckBox(String s, Icon I, boolean state)

Methods

void setSelected(boolean state)

String getText()

void setText(String s)

Event- ItemEvent

4. Radio Buttons

Class- JRadioButton

Constructors-

JRadioButton (String s)

JRadioButton(String s, boolean state)

JRadioButton(Icon i)

JRadioButton(Icon i, boolean state)

JRadioButton(String s, Icon i)

JRadioButton(String s, Icon i, boolean state)

JRadioButton()

To create a button group- ButtonGroup()

Adds a button to the group, or removes a button from the group.

void add(AbstractButton)

void remove(AbstractButton)

5. Combo Boxes

Class- JComboBox

Constructors-

JComboBox()

Methods

void addItem(Object)

Object getItemAt(int)

Object getSelectedItem()

int getItemCount()

Event- ItemEvent

6. List

Constructor-

JList(ListModel)

List models-

1. SINGLE_SELECTION - Only one item can be selected at a time. When the user selects an item, any previously selected item is deselected first.
2. SINGLE_INTERVAL_SELECTION- Multiple, contiguous items can be selected. When the user begins a new selection range, any previously selected items are deselected first.
3. MULTIPLE_INTERVAL_SELECTION- The default. Any combination of items can be selected. The user must explicitly deselect items.

Methods

boolean isSelectedIndex(int)

void setSelectedIndex(int)

void setSelectedIndices(int[])

void setSelectedValue(Object, boolean)

void setSelectedInterval(int, int)

int getSelectedIndex()

int getMinSelectionIndex()

int getMaxSelectionIndex()

int[] getSelectedIndices()

Object getSelectedValue()

Object[] getSelectedValues()

Example

```
listModel = new DefaultListModel();
listModel.addElement("India");
listModel.addElement("Japan");
listModel.addElement("France");
listModel.addElement("Denmark");
list = new JList(listModel);
```

Event- ActionEvent

7. Text classes

All text related classes are inherited from `JTextComponent` class

a. `JTextField`

Creates a text field. The `int` argument specifies the desired width in columns. The `String` argument contains the field's initial text. The `Document` argument provides a custom document for the field.

Constructors-

```
JTextField()  
JTextField(String)  
JTextField(String, int)  
JTextField(int)  
JTextField(Document, String, int)
```

b. `JPasswordField`

Creates a password field. When present, the `int` argument specifies the desired width in columns. The `String` argument contains the field's initial text. The `Document` argument provides a custom document for the field.

Constructors-

```
JPasswordField()  
JPasswordField(String)  
JPasswordField(String, int)  
JPasswordField(int)  
JPasswordField(Document, String, int)
```

Methods-

1. Set or get the text displayed by the text field.

```
void setText(String)  
String getText()
```

2. Set or get the text displayed by the text field.

```
char[] getPassword()
```

3. Set or get whether the user can edit the text in the text field.

```
void setEditable(boolean)  
boolean isEditable()
```

4. Set or get the number of columns displayed by the text field. This is really just a hint for computing the field's preferred width.

```
void setColumns(int);  
int getColumns()
```

5. Get the width of the text field's columns. This value is established implicitly by the font.

```
int getColumnWidth()
```

6. Set or get the echo character i.e. the character displayed instead of the actual characters typed by the user.

```
void setEchoChar(char)  
char getEchoChar()
```

Event- `ActionEvent`

c. JTextArea

Represents a text area which can hold multiple lines of text

Constructors-

JTextArea (int row, int cols)

JTextArea (String s, int row, int cols)

Methods

void setColumns (int cols)

void setRows (int rows)

void append(String s)

void setLineWrap (boolean)

8. Dialog Boxes

Types-

1. Modal- won't let the user interact with the remaining windows of application until first deals with it. Ex- when user wants to read a file, user must specify file name before prg. can begin read operation.
2. Modeless dialog box- Lets the user enters information in both, the dialog box & remainder of application ex- toolbar.

Swing has a JOptionPane class, that lets you put a simple dialog box.

Methods in JOptionPane Class

1. static void showMessageDialog()- Shows a message with ok button.
2. static int showConfirmDialog()- shows a message & gets users options from set of options.
3. static int showOptionDialog- shows a message & get users options from set of options.
4. String showInputDialog()- shows a message with one line of user input.

9. Menu

Creating and Setting Up Menu Bars	
Constructor or Method	Purpose
JMenuBar()	Creates a menu bar.
JMenu add(JMenu)	Creates a menu bar.
void setJMenuBar(JMenuBar) JMenuBar getJMenuBar()	Sets or gets the menu bar of an applet, dialog, frame, internal frame, or root pane.
Creating and Populating Menus	
JMenu() JMenu(String)	Creates a menu. The string specifies the text to display for the menu.
JMenuItem add(JMenuItem) JMenuItem add(Action) JMenuItem add(String)	Adds a menu item to the current end of the menu. If the argument is an Action object, then the menu creates a menu item. If the argument is a string, then the menu automatically creates a JMenuItem object that displays the specified text.

void addSeparator()	Adds a separator to the current end of the menu.
JMenuItem insert(JMenuItem, int) JMenuItem insert(Action, int) void insert(String, int) void insertSeparator(int)	Inserts a menu item or separator into the menu at the specified position. The first menu item is at position 0, the second at position 1, and so on. The JMenuItem, Action, and String arguments are treated the same as in the corresponding add methods.
void remove(JMenuItem) void remove(int) void removeAll()	Removes the specified item(s) from the menu. If the argument is an integer, then it specifies the position of the menu item to be removed.
Implementing Menu Items	
JMenuItem() JMenuItem(String) JMenuItem(Icon) JMenuItem(String, Icon) JMenuItem(String, int)	Creates an ordinary menu item. The icon argument, if present, specifies the icon that the menu item should display. Similarly, the string argument specifies the text that the menu item should display. The integer argument specifies the keyboard mnemonic to use. You can specify any of the relevant VK constants defined in the KeyEvent class. For example, to specify the A key, use KeyEvent.VK_A.
JCheckBoxMenuItem() JCheckBoxMenuItem(String) JCheckBoxMenuItem(Icon) JCheckBoxMenuItem(String, Icon) JCheckBoxMenuItem(String, boolean) JCheckBoxMenuItem(String, Icon, boolean)	Creates a menu item that looks and acts like a check box. The string argument, if any, specifies the text that the menu item should display. If you specify true for the boolean argument, then the menu item is initially selected (checked). Otherwise, the menu item is initially unselected.
JRadioButtonMenuItem() JRadioButtonMenuItem(String) JRadioButtonMenuItem(Icon) JRadioButtonMenuItem(String, Icon) JRadioButtonMenuItem(String, boolean) JRadioButtonMenuItem(Icon, boolean) JRadioButtonMenuItem(String, Icon, boolean)	Creates a menu item that looks and acts like a radio button. The string argument, if any, specifies the text that the menu item should display. If you specify true for the boolean argument, then the menu item is initially selected. Otherwise, the menu item is initially unselected.
void setState(boolean) boolean getState() (in JCheckBoxMenuItem)	Set or get the selection state of a check box menu item.
void setEnabled(boolean)	If the argument is true, enable the menu item. Otherwise, disable the menu item.

Event handling is an important part of GUI based applications. Events are generated by event sources. A mouse click, Window closed, key typed etc. are examples of events.

All java events are sub-classes of java.awt.AWTEvent class.

Java has two types of events:

1. Low-Level Events: Low-level events represent direct communication from user. A low level event is a key press or a key release, a mouse click, drag, move or release, and so on.

Following are low level events.

Event	Description
ComponentEvent	Indicates that a component object (e.g. Button, List, TextField) is moved, resized, rendered invisible or made visible again.
FocusEvent	Indicates that a component has gained or lost the input focus.
KeyEvent	Generated by a component object (such as TextField) when a key is pressed, released or typed.
MouseEvent	Indicates that a mouse action occurred in a component. E.g. mouse is pressed, releases, clicked (pressed and released), moved or dragged.
ContainerEvent	Indicates that a container's contents are changed because a component was added or removed.
WindowEvent	Indicates that a window has changed its status. This low level event is generated by a Window object when it is opened, closed, activated, deactivated, iconified, deiconified or when focus is transferred into or out of the Window.

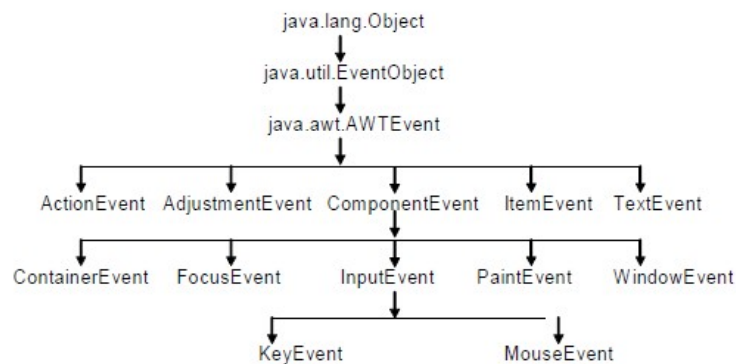
2. High-Level Events: High-level (also called as semantic events) events encapsulate the meaning of a user interface component. These include following events.

Event	Description
ActionEvent	Indicates that a component-defined action occurred. This high-level event is generated by a component (such as Button) when the component-specific action occurs (such as being pressed).
AdjustmentEvent	The adjustment event is emitted by Adjustable objects like scrollbars.
ItemEvent	Indicates that an item was selected or deselected. This high-level event is generated by an ItemSelectable object (such as a List) when an item is selected or deselected by the user.
TextEvent	Indicates that an object's text changed. This high-level event is generated by an object (such as TextComponent) when its text changes.

The following table lists the events, their corresponding listeners and the method to add the listener to the component.

Event	Event Source	Event Listener	Method to add listener to event source
Low-level Events			
ComponentEvent	Component	ComponentListener	addComponentListener()
FocusEvent	Component	FocusListener	addFocusListener()
KeyEvent	Component	KeyListener	addKeyListener()
MouseEvent	Component	MouseListener MouseMotionListener	addMouseListener() addMouseMotionListener()
ContainerEvent	Container	ContainerListener	addContainerListener()
WindowEvent	Window	WindowListener	addWindowListener()
High-level Events			
ActionEvent	Button List MenuItem TextField	ActionListener	addActionListener()
ItemEvent	Choice CheckBox CheckBoxMenuItem List	ItemListener	addItemListener()
AdjustmentEvent	Scrollbar	AdjustmentListener	addAdjustmentListener()
TextEvent	TextField TextArea	TextListener	addTextListener()

Event class hierarchy



Listener Methods:

Methods	Description
ComponentListener	
componentResized(ComponentEvent e)	Invoked when component's size changes.
componentMoved(ComponentEvent e)	Invoked when component's position changes.
componentShown(ComponentEvent e)	Invoked when component has been made visible.
componentHidden(ComponentEvent e)	Invoked when component has been made invisible.
FocusListener	
focusGained(FocusEvent e)	Invoked when component gains the keyboard focus.
focusLost(FocusEvent e)	Invoked when component loses the keyboard focus.
KeyListener	
keyTyped(KeyEvent e)	Invoked when a key is typed.
keyPressed(KeyEvent e)	Invoked when a key is pressed.
keyReleased(KeyEvent e)	Invoked when a key is released.
MouseListener	
mouseClicked(MouseEvent e)	Invoked when a mouse button is clicked (i.e. pressed and released) on a component.
mousePressed(MouseEvent e)	Invoked when a mouse button is pressed on a component.
mouseReleased(MouseEvent e)	Invoked when a mouse button is released on a component.
mouseEntered(MouseEvent e)	Invoked when a mouse enters a component.
mouseExited(MouseEvent e)	Invoked when a mouse exits a component.
MouseMotionListener	
mouseDragged(MouseEvent e)	Invoked when a mouse button is pressed on a component and then dragged.
mouseMoved(MouseEvent e)	Invoked when a the mouse cursor is moved on to a component but mouse button is not pressed.
ContainerListener	
componentAdded(ContainerEvent e)	Invoked when a component is added to the container.
componentRemoved(ContainerEvent e)	Invoked when a component is removed from the container.

WindowListener	
windowOpened(WindowEvent e)	Invoked the first time a window is made visible
windowClosing(WindowEvent e)	Invoked when the user attempts to close the window from the window's system menu.
windowClosed(WindowEvent e)	Invoked when a window has been closed as the result of calling dispose on the window.
windowIconified(WindowEvent e)	Invoked when a window is changed from a normal to a minimized state.
windowDeiconified(WindowEvent e)	Invoked when a window is changed from minimized to normal state.
windowActivated(WindowEvent e)	Invoked when the window is set to be the active window.
windowDeactivated(WindowEvent e)	Invoked when the window is no longer the active window.
ActionListener	
actionPerformed(ActionEvent e)	Invoked when an action occurs.
ComponentListener	
itemStateChanged(ActionEvent e)	Invoked when an item has been selected or deselected by the user.
AdjustmentListener	
adjustmentValueChanged(ActionEvent e)	Invoked when the value of the adjustable has changed.
TextListener	
textValueChanged(ActionEvent e)	Invoked when the value of the text has changed.

Adapter Classes:

All high level listeners contain only one method to handle the high-level events. But most low level event listeners are designed to listen to multiple event subtypes (i.e. the `MouseListener` listens to mouse-down, mouse-up, mouse-enter, etc.). AWT provides a set of abstract "adapter" classes, which implements each listener interface. These allow programs to easily subclass the Adapters and override only the methods representing event types they are interested in, instead of implementing all methods in listener interfaces.

The Adapter classes provided by AWT are as follows:

```
java.awt.event.ComponentAdapter
java.awt.event.ContainerAdapter
java.awt.event.FocusAdapter
java.awt.event.KeyAdapter
java.awt.event.MouseAdapter
java.awt.event.MouseMotionAdapter
java.awt.event.WindowAdapter
```

Applet

Applets are small java programs which are executed and displayed in a java compatible web browser.

Creating an applet

All applets are subclasses of the `java.applet.Applet` class. You can also create an applet by extending the `javax.swing.JApplet` class.

The syntax is:

```
class MyApplet extends Applet
{
//applet methods
}
```

Applet methods:

Method	Purpose
<code>init()</code>	Automatically called to perform initialization of the applet. Executed only once.
<code>start()</code>	Called every time the applet moves into sight on the Web browser to allow the applet to start up its normal operations.
<code>stop()</code>	Called every time the applet moves out of sight on the Web browser to allow the applet to shut off expensive operations.
<code>destroy()</code>	Called when the applet is being unloaded from the page to perform final release of resources when the applet is no longer used
<code>paint()</code>	Called each time the applets output needs to be redrawn.

Running an applet

1. Compile the applet code using `javac`
2. Use the java tool – `appletviewer` to view the applet
(embed the `APPLET` tag in comments in the code)
3. Use the `APPLET` tag in an HTML page and load the applet in a browser

Using appletviewer:

1. Write the HTML `APPLET` tag in comments in the source file.
2. Compile the applet source code using `javac`.
3. Use `appletviewer ClassName.class` to view the applet.

Using browser:

1. Create an HTML file containing the APPLET tag.
2. Compile the applet source code using javac.
3. In the web browser, open the HTML file.

The APPLET tag

< APPLET

[CODEBASE = appletURL]

CODE = appletClassFile

[ALT = alternateText]

[ARCHIVE = archiveFile]

[NAME = appletInstanceName]

WIDTH = pixels

HEIGHT = pixels

[ALIGN = alignment]

[VSPACE = pixels]

[HSPACE = pixels]

>

[< PARAM NAME = AttributeName VALUE = AttributeValue />]

</APPLET>

Attribute	Value	Meaning
align	left right top bottom middle baseline	Specifies the alignment of an applet according to surrounding elements
alt	text	Specifies an alternate text for an applet
archive	URL	Specifies the location of an archive file
code	URL	Specifies the file name of a Java applet
codebase	URL	Specifies a relative base URL for applets specified in the code attribute
height	pixels	Specifies the height of an applet
hspace	pixels	Defines the horizontal spacing around an applet
name	name	Defines the name for an applet (to use in scripts)
vspace	pixels	Defines the vertical spacing around an applet
width	pixels	Specifies the width of an applet

The mandatory attributes are CODE, HEIGHT and WIDTH.

Examples:

1. <applet code=MyApplet width=200 height=200 archive="files.jar">

</applet>

2. <applet code=Simple.class width=100 height=200 codebase="example/">

</applet>

Passing parameters to applets

The PARAM tag allows us to pass information to an applet when it starts running. A parameter is a NAME – VALUE pair. Every parameter is identified by a name and it has a value.

< PARAM NAME = AttributeName VALUE = AttributeValue />

Example:

<APPLET NAME = "MyApplet.class" WIDTH = 100 HEIGHT = 100>

<PARAM NAME = "ImageSource" VALUE = "project/images/">

<PARAM NAME = "BackgroundColor" VALUE = "0xc0c0c0">

<PARAM NAME = "FontColor" VALUE = "Red">

</APPLET>

The Applet can retrieve information about the parameters using the `getParameter()` method.

`String getParameter(String parameterName);`

Example:

`String dirName = getParameter("ImageSource");`

`Color c = new Color(Integer.parseInt(getParameter("BackgroundColor")));`

paint(), repaint() and update()

The `paint()` method redraws the applet. The `repaint()` method is used to force redrawing of the applet. The `update()` method redraws only a portion of the applet.

Assignment – VI

Applets

Practical Assignments

1. Write a program to implement a simple applet that sets the background color to blue, the foreground color to yellow and displays a message that shows the order in which the applet methods are called when an applet starts up.
2. Write a program to display the current date and time using simple applet. Also display greeting message like “Good Morning” or “Good Afternoon” or “Good Evening” depending on the current time in the applet’s status window.
3. Write a program to pass a string as a parameter to applet. Read this parameter and display it on the applet window. Further show all vowels from the read parameter in the applet’s status window.

Assignment Evaluation

0: Not Done	<input type="text"/>	1: Incomplete	<input type="text"/>	2: Late Complete	<input type="text"/>
3: Needs Improvement	<input type="text"/>	4: Complete	<input type="text"/>	5: Well Done	<input type="text"/>
Viva –Voce (0 to 2)	<input type="text"/>				

Signature of the Teacher

Signature of Student

Date ____/____/____

Assignment – VII

AWT Interfaces

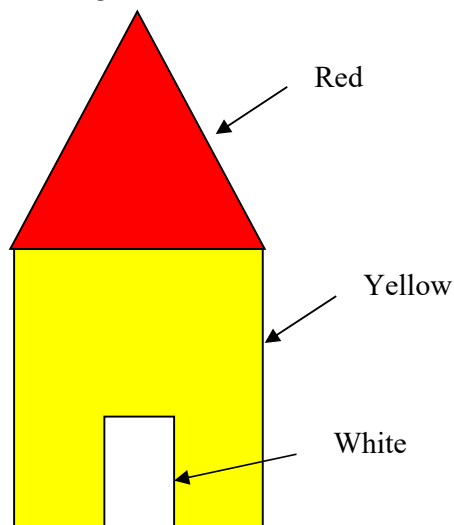
Practical Assignments

1. Write a program to design the following screen.

Mouse X	Mouse X coordinate
Mouse Y	Mouse Y coordinate

Where Mouse X, Mouse Y, Mouse X coordinate and Mouse Y Coordinate are Labels.

2. Write a program to design following screen.



3. Write a program to obtain and display the names of the available font families on multiple lines of text.

Assignment Evaluation

0: Not Done ☐ 1: Incomplete ☐ 2: Late Complete ☐

3: Needs Improvement ☐ 4: Complete ☐ 5: Well Done ☐

Viva –Voce (0 to 2) ☐

Signature of the Teacher

Signature of Student

Date ____/____/____

Assignment – VIII

Swing Interfaces




Practical Assignments

1. Write a program to design the following screen of number conversion.

Decimal	<input type="text"/>
Binary	Result in Binary
Octal	Result in Octal
Hexadecimal	Result in Hexadecimal
<input type="button" value="Convert"/>	<input type="button" value="Exit"/>

Where Decimal, Binary, octal, Hexadecimal, Result in Binary, Result in Octal and Result in Hexadecimal are label. Blank rectangle is textbox. Convert and Exit are buttons.

2. Write a program to design the following screen

Font	Font Style	Size
<input type="text" value="Font Family"/> 	<input type="radio"/> Bold	<input type="text" value="8"/> 
	<input type="radio"/> Italic	<input type="text" value="9"/>
		<input type="text" value="10"/> 
<input type="text" value="Hello Everyone"/>	<input type="button" value="Apply"/>	<input type="button" value="Exit"/>

Where Font, Font Style and Size are label. Font Family is a combo box. Bold and Italic are radio or option button. Numbers is a list. Hello Everyone, is text box. Apply and Exit are buttons.

Assignment Evaluation

0: Not Done	<input type="text"/>	1: Incomplete	<input type="text"/>	2: Late Complete	<input type="text"/>
3: Needs Improvement	<input type="text"/>	4: Complete	<input type="text"/>	5: Well Done	<input type="text"/>
Viva –Voce (0 to 2)	<input type="text"/>				

Signature of the Teacher

Signature of Student

Date ____/____/____

Assignment – IX

Event Listeners

Practical Assignments

1. Write a program to implement question number 1 of assignment VII which will displays the current coordinates of mouse in the label Mouse X coordinate and Mouse Y coordinate.
2. Write a program to implement question number 1 of assignment VIII and display the appropriate number conversion results in the label after clicking Convert button. Use Exit button to terminate the application.
3. Write a program to implement question number 2 of assignment VIII and apply the font, font style and size effect on the text “Hello Everyone” after clicking Apply button. Use Exit button to terminate the application.

Assignment Evaluation

0: Not Done 1: Incomplete 2: Late Complete

3: Needs Improvement 4: Complete 5: Well Done

Viva –Voce (0 to 2)

Signature of the Teacher

Signature of Student

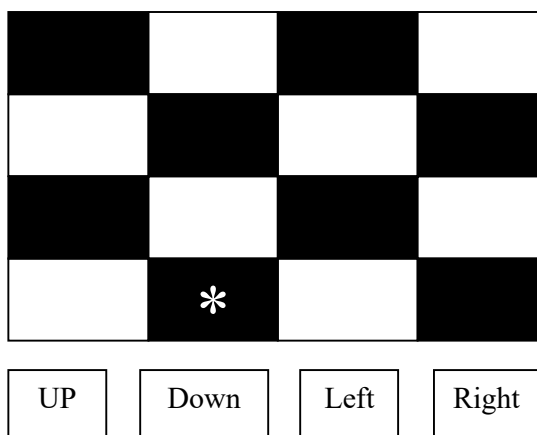
Date ____/____/____

Assignment – X

Applet / Swing Interface / Adapter class

Practical Assignments

1. Write a program to demonstrate keyboard input. It displays keystrokes to the applet window and shows the status of each key in the applet's status window.
2. Create an application in Java using swing that will move star towards up, down, left, and right from its current position. Use appropriate layout managers or your own layout. The color of * should change from black to white and vice-versa depending on the box it is present. Display appropriate message if movement try to cross the boundary. Design the screen as shown:



3. Write a program to handle mouse clicked, mouse motion, mouse dragged events using adapter class. Display the status of mouse in the applet's status window.

Assignment Evaluation

0: Not Done 1: Incomplete 2: Late Complete

3: Needs Improvement 4: Complete 5: Well Done

Viva –Voce (0 to 2)

Signature of the Teacher

Signature of Student

Date ____/____/____

References

Books:

1. Cay S. Horstmann, Gary Cornell, Core Java Volume-I- Fundamentals, Ninth Edition, 2015.
2. Ivan Bayross, Commercial web development using java 2.0, BPB, 2007.
3. Steven Horlzner , Java 2 programming black books, 2006.
4. Herbert Schildt(5th edition), Complete reference Java, 2002.

Web References:

1. <https://www.javatpoint.com/java>
2. <https://www.tutorialspoint.com/java>
3. <https://www.studytonight.com/java>
4. <https://www.w3schools.com/java>
5. https://www.tutorialspoint.com/javamail_api
