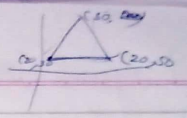
(2) ~~go~~ Font.

```
import java.applet.*;
import java.awt.*;
~~etc~~ public class font extends Applet
{
String msg;
Font f;
public void init()
{
f = new Font("Times New Roman",
        ~~FONT~~ Font.BOLD, 30);
setFont(f);
}
public void paint(Graphics g)
{
g.drawString("hello", 200, 200);
f = g.getFont();
msg = f.getName();
g.drawString("FONT NAME IS" + msg,
        300, 300);
~~g.drawPolygon(x, y, num);~~
~~g.fillPolygon(~~
}
}
/*<applet code="font.java" width=500
heigh= 500></applet>*/
```



(3) Polygon.

```
import java.applet.*;
import java.awt.*;
public class polygon extends
    Applet
{
int num = 3;
int x[] = { 50, 100, 270, 190};
int y[] = { 100, 280, 470, 290};
~~public void int()~~
  public void paint(Graphics g)
{
g.drawPolygon(x, y, num);
g.fillPolygon(x, y, num);
}
}
/*<applet code="polygon.java"
width= 500 heig ht= 500></applet>
*/
```

Palindrome:-

```
import java.util.*;
class palind
{
public static void main (String arg
{
int res = 0, remainder, num, num1;
Scanner sc = new Scanner (System.in);
num = sc.nextInt();
num1 = num;
while (num>0)
{
remainder = num % 10;
res = res * 10 + remainder;
num = num / 10;
}
System.out.println("Palindrome" + res);
if (num1 == res)
System.out.println("Entered number is palindrome:");
else
System.out.println("Entered number is not a palindrome:");
}
}
}
```

Prime Number prg:-

```
import java.util.*;
class prime
{
public static void main (String arg[])
{
int num;
boolean flag = true;
Scanner sc = new Scanner (System.in);
System.out.println("Enter num");
num = sc.nextInt();
for (int i = 2; i<num/2; i++)
{
if (num % i == 0)
{
flag = false;
break;
}
System.out.println("not prime");
}
if (flag == true)
System.out.println("prime num");
else
System.out.println("Not prime");
}
}
}
```

Fibonnanei Series.

```
class fibo
{
public static void main (String args[])
{
int num1 = 0 ; num2 = 1; ans;
system.out.println( num1);
while (num2 < 100)
{
ans = num1 + num2;
num1 = num2;
num2 = ans;
}
}
}
```

123 % 10

3+2+1

1+2+3

456

4+5+6

Sum of digit of number entered by the user.

```
import java.util.*;
class Sum
{
public static void main (String args[])
{
Sc int res=0, remainder, num;
```

Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a web document. The applet states include:

- Born or initialization state
- Running state
- Idle state
- Dead or destroyed state

Initialization state: Applet enters the initialization state when it is first loaded. This is done by calling the init() method of Applet class. At this stage the following can be done:

- Create objects needed by the applet
- Set up initial values
- Load images or fonts
- Set up colors

Initialization happens only once in the life time of an applet.

```
public void init()
 {
//implementation
 }
```

Running state: applet enters the running state when the system calls the start() method of Applet class. This occurs automatically after the applet is initialized. start() can also be called if the applet is already in idle state. start() may be called more than once. start() method may be overridden to create a thread to control the applet.

```
public void start()
 {
 //implementation
 }
```

Idle or stopped state: an applet becomes idle when it is stopped from running. Stopping occurs automatically when the user leaves the page containing the currently running applet. stop() method may be overridden to terminate the thread used to run the applet.

```
 public void stop()
 {
//implementation
 }
```

Dead state: an applet is dead when it is removed from memory. This occurs automatically by invoking the destroy method when we quit the browser. Destroying stage occurs only once in the lifetime of an applet. destroy() method may be overridden to clean up resources like threads.

```
 public void destroy()
 {
 //implementation
 }
```

public class Test {
public static void main(String args[]) {
int i = 100;
long l = i; // no explicit type casting require
float f = l; // no explicit type casting required
System.out.println ("Int value " + i);
System.out.println ("Long value " + l);
System.out.println ("Float value " + f);
}}
	Output:
	
	Int value 100
	Long value 100
	Float value 100.0

2. Narrowing or Explicit type casting

- When you are assigning a larger type value to a variable of smaller type. Then you need to perform explicit type casting.

public class Test
{
public static void main(String args[]) {
double d = 100.04;
long l = (long) d; // explicit type casting required
int I = (int) l; // explicit type casting required
 	
System.out.println ("Double value " + d);
System.out.println ("Logn value " + l);
System.out.println ("Int value " + I);
 	
}
}
	Output:
	
	Double value 100.04
	Long value 100
	Int value 100

Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub-topics
	settings.	size, style, font methods: getFamily(), getFont(), getFontname (), getSize(), getStyle(), getAllFonts() and get available font family name() of the graphics environment class.
Unit –VI Managing Input /Output/ Files in Java	6a. Use I/O stream classes in a program to solve the given problem. 6b. Write programs for reading and writing character streams to and from the given files. 6c. Write programs for reading and writing bytes to and from the given files. 6d. Write program to demonstrate use of primitive Data types with the specified stream.	6.1 Introduction and Concept of Streams. 6.2 Stream Classes. 6.3 Byte Stream Classes: Input Stream Classes, Output Stream Classes. 6.4 Character Stream Classes, Using streams. 6.5 Using File Class: I/O Exceptions, Creation of Files, Reading/Writing characters, Reading/Writing Bytes, Handling Primitive Data types.

***Note**: To attain the CO s and competency, above listed UOs need to be undertaken to achieve the 'Application Level' of Bloom's 'Cognitive Domain Taxonomy'*

9. SUGGESTED SPECIFICATION TABLE FORQUESTION PAPER DESIGN

Unit No.	Unit Title	Teaching Hours	Distribution of Theory Marks			
			R Level	U Level	A Level	Total Marks
I	Basic Syntactical constructs in Java	06	02	04	04	10
II	Derived Syntactical Constructs in Java	10	02	06	10	18
III	Inheritance, Interface and Package	10	02	04	06	12
IV	Exception Handling and Multithreading	08	02	04	06	12
V	Java Applets and Graphics Programming	08	02	04	04	10
VI	Managing Input/Output/Files in Java	06	02	02	04	08
Total		**48**	**12**	**24**	**34**	**70**

***Legends:** R=Remember, U=Understand, A=Apply and above (Bloom's Revised taxonomy)*
***Note**: This specification table provides general guidelines to assist students for their learning and to teachers to teach and assess students with respect to attainment of UOs. The actual distribution of marks at different taxonomy levels (of R, U and A) in the question paper may vary from above table*

10. SUGGESTED STUDENT ACTIVITIES

Other than the classroom and laboratory learning, following are the suggested student-related ***co-curricular*** activities which can be undertaken to accelerate the attainment of the various outcomes in this course: Students should conduct following activities in group and prepare reports of about 5 pages for each activity, also collect/record physical evidences for their (student's) portfolio which will be useful for their placement interviews:

a. Prepare journals based on practical performed in laboratory.
b. Follow coding standards.
c. Develop variety of programs to improve the logical skills.
d. Develop Application oriented real world programs.
e. Prepare power point presentation or animation for understanding different Object Oriented Concepts.

... avoid polling **there are three in-built methods** that take part in inter-thread communication -

notify()	If a particular thread is in the sleep mode then that thread can be resumed using the notify call.
notifyall()	This method resumes all the threads that are in suspended state.
wait()	The calling thread can be send into a sleep mode.

4.19 Deadlock

Deadlock is a situation in which two or more threads are waiting for object lock which is hold by some another thread.

For example - Consider thread A is waiting for object obj2 which is locked by thread B and thread B is waiting for object obj1 which is locked by thread A. This situation is described as deadlock.

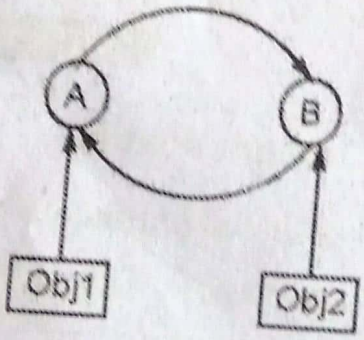


Fig. 4.19.1 : Deadlock

Following Java program shows the deadlock situation

Java Program[DeadLockDemo.java]

```
public class DeadLockDemo
{
```

Board Questions

1. *Explain thread priority and method to get and set priority values.*

   MSBTE : Summer 15, Marks 4

2. *What is thread priority ? How thread priority are set and changed ? Explain with example.*

   MSBTE : Summer 16, Marks 8

3. *What is thread priority ? Write default priority values and methods to change them.*

   MSBTE : Summer 17, Marks 4

4.17 Synchronization

MSBTE.: Winter 16, Marks 4

- When two or more threads need to access shared memory, then there is some way to ensure that the access to the resource will be by only one thread at a time. The process of ensuring one access at a time by one thread is called synchronization. The synchronization is the concept which is based on monitor. Monitor is used as mutually exclusive lock or mutex. When a thread owns this monitor at a time then

e) Write a program to count number of words from a text file using

Q.5) Attempt any TWO of the following. **12 Marks**

a) Write a step to declare and define two and three dimensional arrays of a class.

```
result = (num1>num2) ? num1 : num2
System.out.println("Largest is:"+
                    result);
    }
}
```

```
Student (int r, String n)
  {
  rollno = r;
  name = n;
  }
Student (int r)
  {
  rollno = r;
  }
void display ()
  {
System.out.println("Rollno is:" +
                    rollno);
System.out.println("Name is:" +
```

```
class sdemos
{
 public static void main
      (String args[])
  {
Student s= new student (10,"RAJ");
   s.display();
student s= new student(20,"ABC");
   s.display();
  }
 }
```

(*) Write a program to find largest between two numbers using "?:" operator.

→
```
import java.util.*;
class large
{
 public static void main
      (String args[])
  {
 int num1, num2, result;
 Scanner sc= new Scanner
      (System.in);
System.out.println("Enter num1
      & num2:");
num1= sc.nextInt();
num2= sc.nextInt();
```

- Java allows its programmers to initialize a variable at run time also. Initializing a variable at run time is called dynamic initialization.

```
Double sr = Math.sqrt(100);
```

Syllabus Topic : Array and Strings

1.11 Array and Strings

- Array is a group of elements with similar data types.
- Array of characters is known as String.
- This topic we are going to discuss in detail in next chapter.

Syllabus Topic : Scope of Variable

1.12 Scope of Variable

➔ **(MSBTE - W-16, S-18)**

Q. 1.12.1	What is scope of variable? Give example of class variable, instance variable and local variable. (Refer section 1.12)	W-16, 4 Marks
Q. 1.12.2	State & explain scope of variable with an example. (Refer section 1.12)	S-18, 4 Marks

- **Scope** of a variable refers to; areas or sections of a program in which the variable can be accessible and **lifetime** of a variable refers to how long the variable stays alive in the memory.
- General convention for a variable's scope is, it is accessible only within the block in which it is declared. A block begins with a left curly brace { and ends with a right curly brace }.
- There are three types of variables :

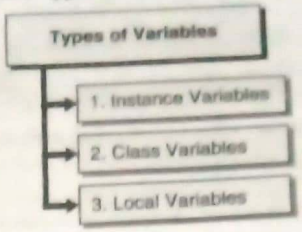


Fig. 1.12.1 : Types of Variables

➔ **1. Instance Variables**

- A variable which is declared inside a class and outside all the methods and blocks is an instance variable.
- General **scope of an instance variable** is throughout the class except in static methods.

➔ **2. Class Variables**

- A variable which is declared inside a class, outside all the blocks and is marked *static* is known as a class variable.
- General **scope of a class variable** is throughout the class.

➔ **3. Local Variables**

- All other variables which are not instance and class variables are treated as local variables including the parameters in a method.
- **Scope of a local variable** is within the block in which it is declared.

Syllabus Topic : Typecasting and Standard Default Values

1.13 Typecasting and Standard Default Values

A) Typecasting

➔ **(MSBTE - W-14, S-18)**

Q. 1.13.1	What do mean by typecasting? When it is needed? (Refer section 1.13)	W-14, 4 Marks

- Java allows its programmers to initialize a variable at run time also. Initializing a variable at run time is called dynamic initialization.

```
Double sr = Math.sqrt(100);
```

Syllabus Topic : Array and Strings

1.11 Array and Strings

- Array is a group of elements with similar data types.
- Array of characters is known as String.
- This topic we are going to discuss in detail in next chapter.

Syllabus Topic : Scope of Variable

1.12 Scope of Variable

➔ **(MSBTE - W-16, S-18)**

Q. 1.12.1 What is scope of variable? Give example of class variable, instance variable and local variable.
(Refer section 1.12) **W-16, 4 Marks**

Q. 1.12.2 State & explain scope of variable with an example.
(Refer section 1.12) **S-18, 4 Marks**

- **Scope** of a variable refers to; areas or sections of a program in which the variable can be accessible and **lifetime** of a variable refers to how long the variable stays alive in the memory.
- General convention for a variable's scope is, it is accessible only within the block in which it is declared. A block begins with a left curly brace { and ends with a right curly brace }.
- There are three types of variables :

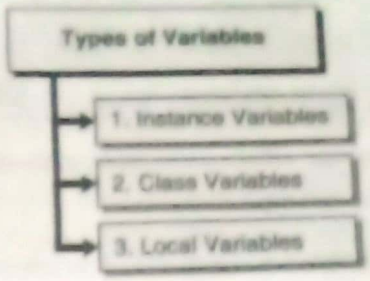


Fig. 1.12.1 : Types of Variables

➔ **1. Instance Variables**

- A variable which is declared inside a class and outside all the methods and blocks is an instance variable.
- General **scope of an instance variable** is throughout the class except in static methods.

➔ **2. Class Variables**

- A variable which is declared inside a class, outside all the blocks and is marked *static* is known as a class variable.
- General **scope of a class variable** is throughout the class.

➔ **3. Local Variables**

- All other variables which are not instance and class variables are treated as local variables including the parameters in a method.
- **Scope of a local variable** is within the block in which it is declared.

Syllabus Topic : Typecasting and Standard Default Values

1.13 Typecasting and Standard Default Values

A) Typecasting

➔ **(MSBTE - W-14, S-18)**

Q. 1.13.1 What do mean by typecasting? When it is needed? (Refer section 1.13) **W-14, 4 Marks**

[illegible]

Syllabus Topic : Array and Strings

[illegible] Array and Strings

- [illegible]
- [illegible]
- [illegible]

Syllabus Topic : Scope of Variable

[illegible] Scope of Variable

(2021) (6-6) 2-11

Q.1.1.1 [illegible] **Scope of a variable** [illegible]

[illegible]

Lifetime of a variable refers to time [illegible]

[illegible]

Syllabus Topic : Typecasting and Standard Default Values

1.13 Typecasting and Standard Default Values

A) Typecasting :

(2021) (6-6) 2-12

Q.1.1.2 [illegible]

[illegible]

[illegible]

Syllabus Topic : Typecasting and Standard Default Values

1.13 Typecasting and Standard Default Values

A) Typecasting :

(2021) (6-6) 2-12

Q.1.1.2 What is meant by typecasting? Write it in [illegible]

[illegible]

- All these states are represented by five methods means there exists 5 states represented by 5 methods.
- These methods are automatically called by the browser for smooth execution of the applet.
- The applet life cycle methods are members of **java.applet.Applet** class except **paint()** method.
- The paint() method is member of **java.awt.Component** class which is an indirect super class of Applet.
- The life cycle of an Applet is as shown in Fig. 5.4.1.

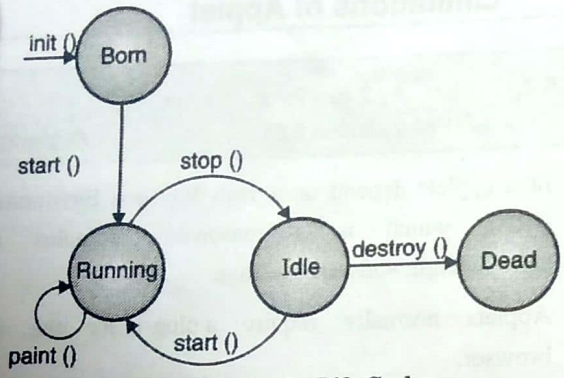


Fig. 5.4.1 : Applet Life Cycle

1. **Init() :** This method signifies the beginning of applet's life cycle. It is used to initialize the Applet. In this method the object of Applet class is created. As this method is called before all the remaining methods, it is used to initialize the variables, instantiate objects, setting background as well as foreground colors in GUI etc.

   Invocation : This method is invoked only once at the beginning of an Applet. Consider a website made up of web-pages (applets), when we visit any webpage then first its init method is called.

2. **Start() :** After init(), the start() method is invoked. It starts the execution of Applet. In this state, it is considered that the applet becomes active.

   Invocation : This method is repeatedly called in the life of an Applet. This method is invoked whenever user revisits any webpage (Applet). It also executes whenever the applet is restored, maximized or user is shifting from one tab to another tab in the window of browser.

3. **Paint() :** This method is used to create Applet's GUI such as a colored background, drawing different shapes and printing messages etc.

   Invocation : This method is called only once by the browser. But user can recall it by using repaint() method.

4. **Stop() :** This method is used to stop execution of an applet temporary.

   Invocation : This method is repeatedly called in the life of an Applet. Whenever user shifts to another webpage (Applet), this method is invoked. It is invoked when Applet is stopped or browser is minimized.

5. **Destroy()**

   This method indicates the end of applet's life cycle. Here user can write the cleanup code. It removes the applet object from memory.

   Invocation : This method is called only once by the browser when the applet window is closed or when the browser or the tab of browser containing the webpage is closed.

5.5 Applet Class

Q. 5.5.1 Write a note on Applet Class. (Refer section 5.5) (4 Marks)

- Java provides class Applet in the java.applet packa which contains several methods to handle t functionality of an Applet.
- The hierarchical structure for applets class is follows :

java.lang.Object
⇓
java.awt.Component
⇓
java.awt.Container
⇓
java.awt.Panel
⇓
java.applet.Applet

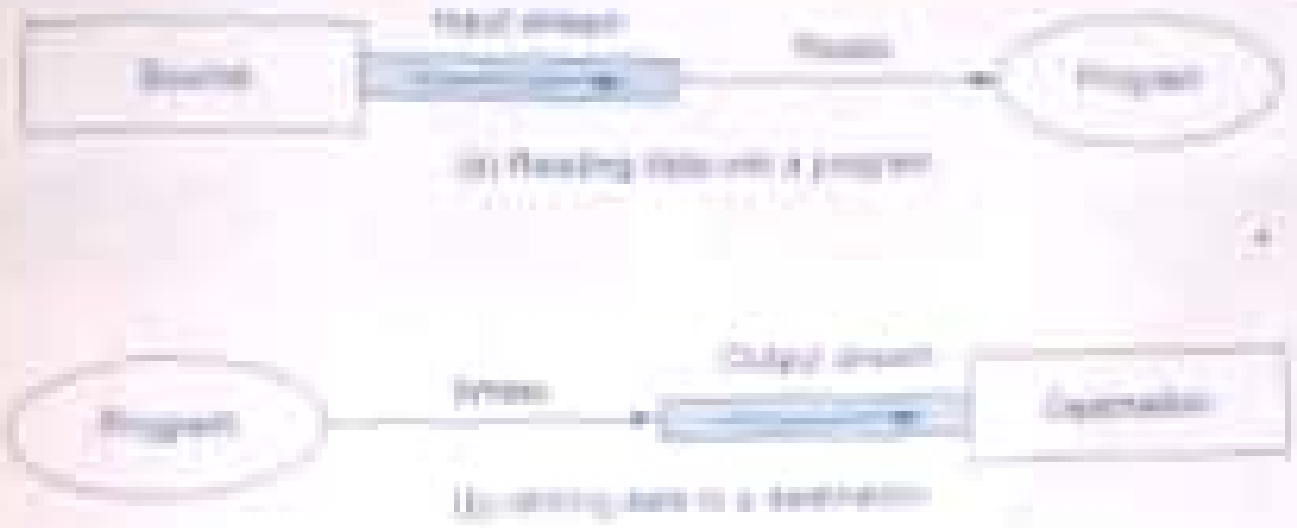


Fig. 10.4 Using input and output streams

- Reserved Keywords
- Identifiers
- Literals
- Operators
- Separators

Java Character Set:

The smallest unit of Java language are the characters used in writing Java options. These characters are defined by the Unicode character set, an emerging standard that tries to cover the needs of a large number of written languages.

The Unicode is a 16-bit character coding system and currently supports most languages of the world. Characters defined from 24 languages from Asia, Europe, Middle East, Africa and Americas including India. Currently, most of us are only using the basic ASCII characters, which range from 0 to 127.

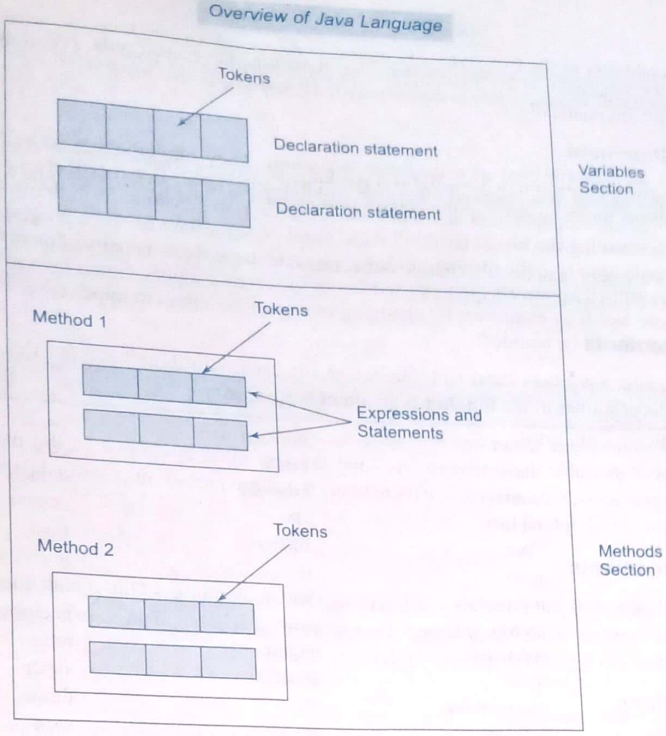


Fig. 3.3 *Elements of Java class*

In simple terms, a Java program is a collection of tokens, comments and white spaces. Java language includes five types of tokens. They are:

- Reserved Keywords
- Identifiers
- Literals
- Operators
- Separators

Java Character Set

The smallest units of Java language are the characters used to write Java tokens. These characters are defined by the *Unicode* character set, an emerging standard that tries to create characters for a large number of scripts worldwide.

The Unicode is a 16-bit character coding system and currently supports more than 34,000 defined characters derived from 24 languages from America, Europe, Middle East, Africa and Asia (including India). However, most of us use only the basic ASCII characters, which include letters, digits and