AKTU

B. Tech II-Year

Gateway Classes

Semester -IV

CS IT & Allied Branches

BCS403-Object Oriented Programming with Java

UNIT-2 :Exception Handling

Gateway Series for Engineering

- Topic Wise Entire Syllabus
- Long - Short Questions Covered
- AKTU PYQs Covered
- DPP
- Result Oriented Content

Download App

For Full Courses including Video Lectures

AKTU

B. Tech II-Year

Gateway Classes

BCS403- Object Oriented Programming with Java

Unit-2

Introduction to Exception Handling

Syallbus

Exception Handling: The Idea behind Exception, Exceptions & Errors, Types of Exception, Control Flow in Exceptions, JVM Reaction to Exceptions, Use of try, catch, finally, throw, throws in Exception Handling, In-built and User Defined Exceptions, Checked and Un-Checked Exceptions.

Input /Output Basics: Byte Streams and Character Streams, Reading and Writing File in Java

. **Multithreading:** Thread, Thread Life Cycle, Creating Threads, Thread Priorities, Synchronizing Threads, Inter-thread Communication.

Download App

For Full Courses including Video Lectures

AKTU 4th SEM

CS IT & Allied Branches

OOPs with JAVA

UNIT-2 Lec-1

Enhance Your Skills

All Universities

Dr. Sandeep Kumar Gupta

What We will Discuss Today?

- What is Exception Handling?
- Why its Required?
- Difference between Errors and Exceptions
- Try, catch, finally, throw and throws
- Library Defined Exceptions
- Custom Exceptions
- Difference b/w Checked and Unchecked Exceptions

Types of Errors in a Program

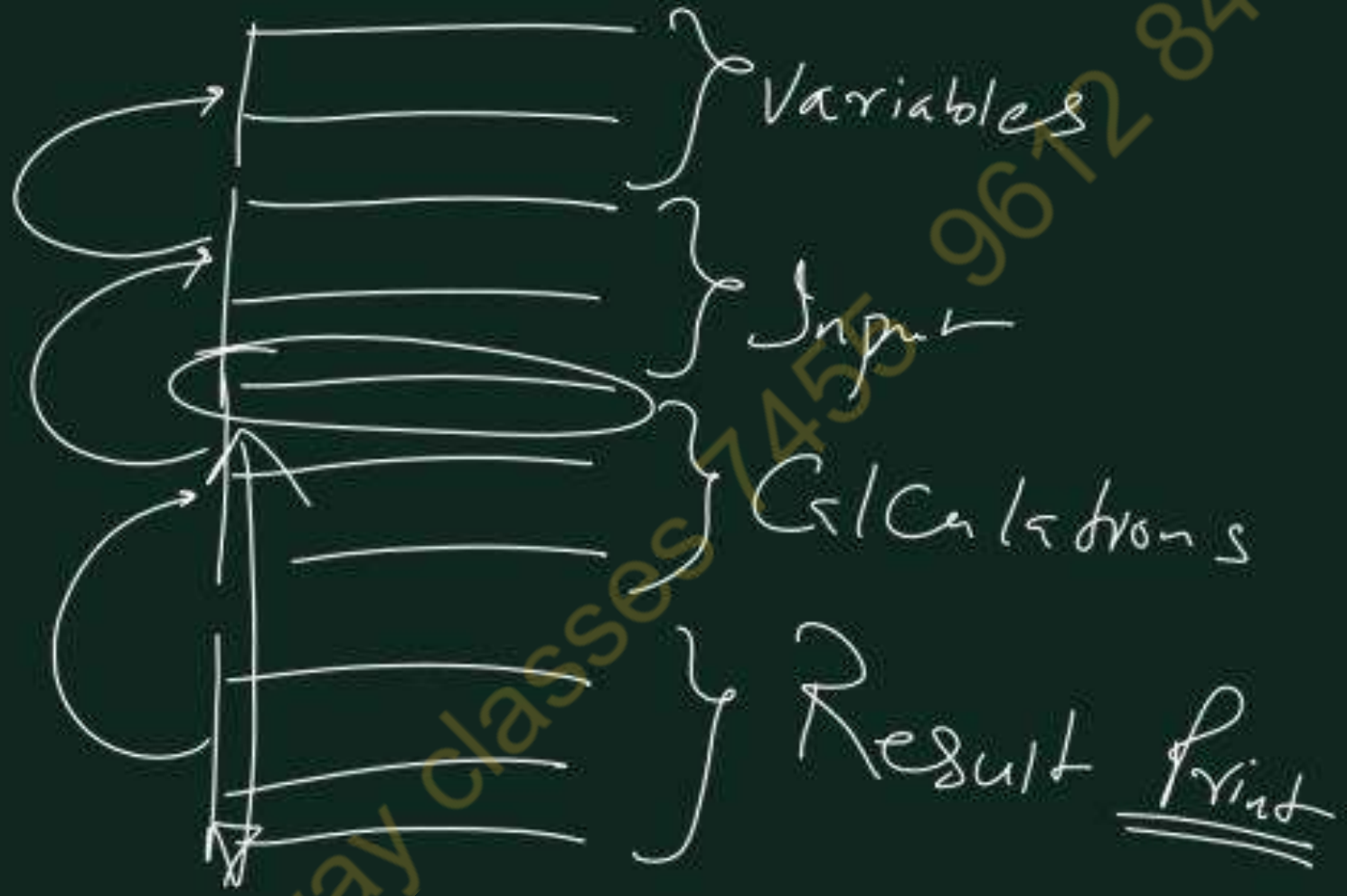
- Syntax Error

```
class A
{
        public static void main(String[] ar)
        {
             system.out.print("Hello)
        }
}
```

- Logical Error

```
class A
{
        public static void main(String[] ar)
        {
                ..........
                while(x>0)
                {
                        ..........
                        ..........
                }
                ..........
        }
}
```

- Runtime Errors



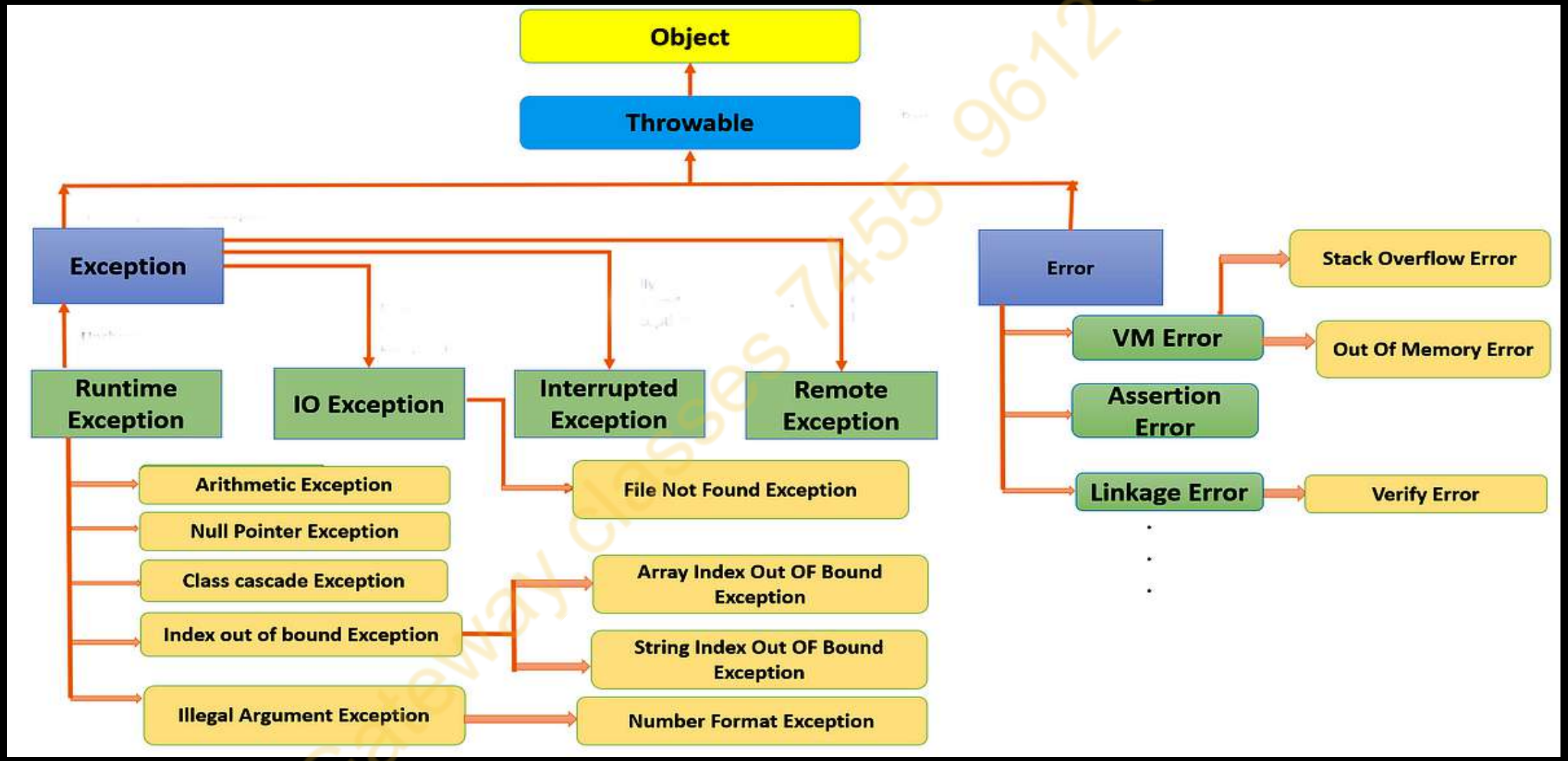
Runtime Error/Exceptions

```
import java.util.Scanner;
class A
{
        public static void main(String[] ar)
        {
                Scanner sc=new Scanner(System.in);
                int[] arr={23,45,8,54,23,56};
                int i,a,ans=0;
                System.out.println("Enter Index:");
                i=sc.nextInt();
                System.out.println("Enter a Value:");
                a=sc.nextInt();
                ans=arr[i]/a;
                System.out.println("Ans="+ans);

        }
}
```

i = 3
a = 4
arr[3]/4
54/4

Exception Hierarchy



```
1.  import java.util.Scanner;
2.  class A
3.  {
4.      public static void main(String[] ar)
5.      {
6.          Scanner sc=new Scanner(System.in);
7.          int[] arr={23,45,8,54,23,56};
8.          int i,a,ans=0;
9.          System.out.println("Enter Index:");
10.         try{
11.             i=sc.nextInt();
12.             System.out.println("Enter a Value:");
13.             a=sc.nextInt();
14.             ans=arr[i]/a;
15.         }catch(Exception ee)
16.         {
17.             System.out.println("Some Error Occurred");
18.         }
19.         System.out.println("Ans="+ans);
20.     }
21. }
```

```
1.  import java.util.Scanner;
2.  class A
3.  {
4.      public static void main(String[] ar)
5.      {
6.          Scanner sc=new Scanner(System.in);
7.          int[] arr={23,45,8,54,23,56};
8.          int i,a,ans=0;
9.          System.out.println("Enter Index:");
10.         try{
11.             i=sc.nextInt();
12.             System.out.println("Enter a Value:");
13.             a=sc.nextInt();
14.             ans=arr[i]/a;
15.         }catch(ArrayIndexOutOfBoundsException ee)
16.         {
17.             System.out.println("Array Index Problem");
18.         }catch(ArithmeticException ee)
19.         {
20.             System.out.println("Second Value Can't be zero");
21.         }
22.         System.out.println("Ans="+ans);
23.     }
24. }
```

1) try — catch

2) try — finally

3) try — catch — finally

4) try — catch — catch — — —

5) try — catch — catch — —

Nested

try

catch

finally

throw

throws

finally

MyException e = new MyException();

throw e;

throws

Class A
{
void f1() throws AE, IME, ...
{
}
}

Class B
{
psvm()
{
A a = new A();
a.f1();
}
}

```
1.  import java.util.Scanner;
2.  class A
3.  {
4.      public static void main(String[] ar)
5.      {
6.          Scanner sc=new Scanner(System.in);
7.          int[] arr={23,45,8,54,23,56};
8.          int i,a,ans=0;
9.          System.out.println("Enter Index:");
10.         try{
11.             i=sc.nextInt();
12.             System.out.println("Enter a Value:");
13.             a=sc.nextInt();
14.             ans=arr[i]/a;
15.         }catch(ArrayIndexOutOfBoundsException ee)
16.         {
17.             System.out.println("Array Index Problem");
18.         }catch(ArithmeticException ee)
19.         {
20.             System.out.println("Second Value Can't be zero");
21.         }finally
22.         {
23.             System.out.println("Finally Block");
24.         }
25.         System.out.println("Ans="+ans);
26.     }
27. }
```

Exception Handling

```
import java.util.*;
class A41
{
	public static void main(String[] ar)
	{
		Scanner sc=new Scanner(System.in);
		int[] arr={23,45,67,78,34,12,47,78};
		int i,a,ans=0;
		System.out.println("Enter Index:");
		try{
			i=sc.nextInt();
			System.out.println("Enter Value:");
			a=sc.nextInt();
			ans=arr[i]/a;
		}catch(Exception ee)
		{
			System.out.println("Some Error Occurred");
		}
		System.out.println("Ans="+ans);
	}
}
```

```
import java.util.*;
class A42
{
	public static void main(String[] ar)
	{
		Scanner sc=new Scanner(System.in);
		int[] arr={23,45,67,78,34,12,47,78};
		int i,a,ans=0;
		System.out.println("Enter Index:");
		try{
			i=sc.nextInt();
			System.out.println("Enter Value:");
			a=sc.nextInt();
			ans=arr[i]/a;
		}catch(ArrayIndexOutOfBoundsException ee)
		{
```

```
                System.out.println("Index should be <8");
            }catch(InputMismatchException ee)
            {
                System.out.println("Enter numbers Only");
            }catch(Exception ee)
            {
                System.out.println("Some error");
            }
            System.out.println("Ans="+ans);
        }
}
```

```
import java.util.*;
class A43
{
    public static void main(String[] ar)
    {
        Scanner sc=new Scanner(System.in);
        int[] arr={23,45,67,78,34,12,47,78};
        int i,a,ans=0;
        System.out.println("Enter Index:");
        try{
            i=sc.nextInt();
            System.out.println("Enter Value:");
            a=sc.nextInt();
            ans=arr[i]/a;
        }catch(ArrayIndexOutOfBoundsException ee)
        {
            System.out.println("Index should be <8");
        }catch(InputMismatchException ee)
        {
            System.out.println("Enter numbers Only");
        }finally
        {
            System.out.println("Finally Block");
        }
        System.out.println("Ans="+ans);
    }
}
```

```
import java.util.*;
class InvalidAgeException extends RuntimeException
{
	public String toString()
	{
		return "InvalidAgeException:Valid age should be >18 and <50";
	}
}
class A44
{
	public static void main(String[] ar)
	{
		Scanner sc=new Scanner(System.in);
		int age;
		System.out.print("Enter Age:");
		age=sc.nextInt();
		if(age<18 || age>50)
		{
			System.out.println("Invalid Age");
			InvalidAgeException ee=new InvalidAgeException();
			throw ee;
		}
		System.out.println("A1");
		System.out.println("A2");
		System.out.println("A3");
		System.out.println("A4");
	}
}
```

```
import java.util.*;
class InvalidAgeException extends Exception
{
	public String toString()
	{
		return "InvalidAgeException:Valid age should be >18 and <50";
	}
}
```

```
class A45
{
	public static void main(String[] ar)
	{
		Scanner sc=new Scanner(System.in);
		int age;
		System.out.print("Enter Age:");
		age=sc.nextInt();
		if(age<18 || age>50)
		{
			System.out.println("Invalid Age");
			InvalidAgeException ee=new InvalidAgeException();
			try{
				throw ee;
			}catch(Exception e)
			{
			}
		}
		System.out.println("A1");
		System.out.println("A2");
		System.out.println("A3");
		System.out.println("A4");
	}
}
```

```
import java.util.*;
class InvalidAgeException extends Exception
{
	public String toString()
	{
		return "InvalidAgeException:Valid age should be >18 and <50";
	}
}
class A45
{
	public static void main(String[] ar) throws InvalidAgeException
	{
		Scanner sc=new Scanner(System.in);
		int age;
```

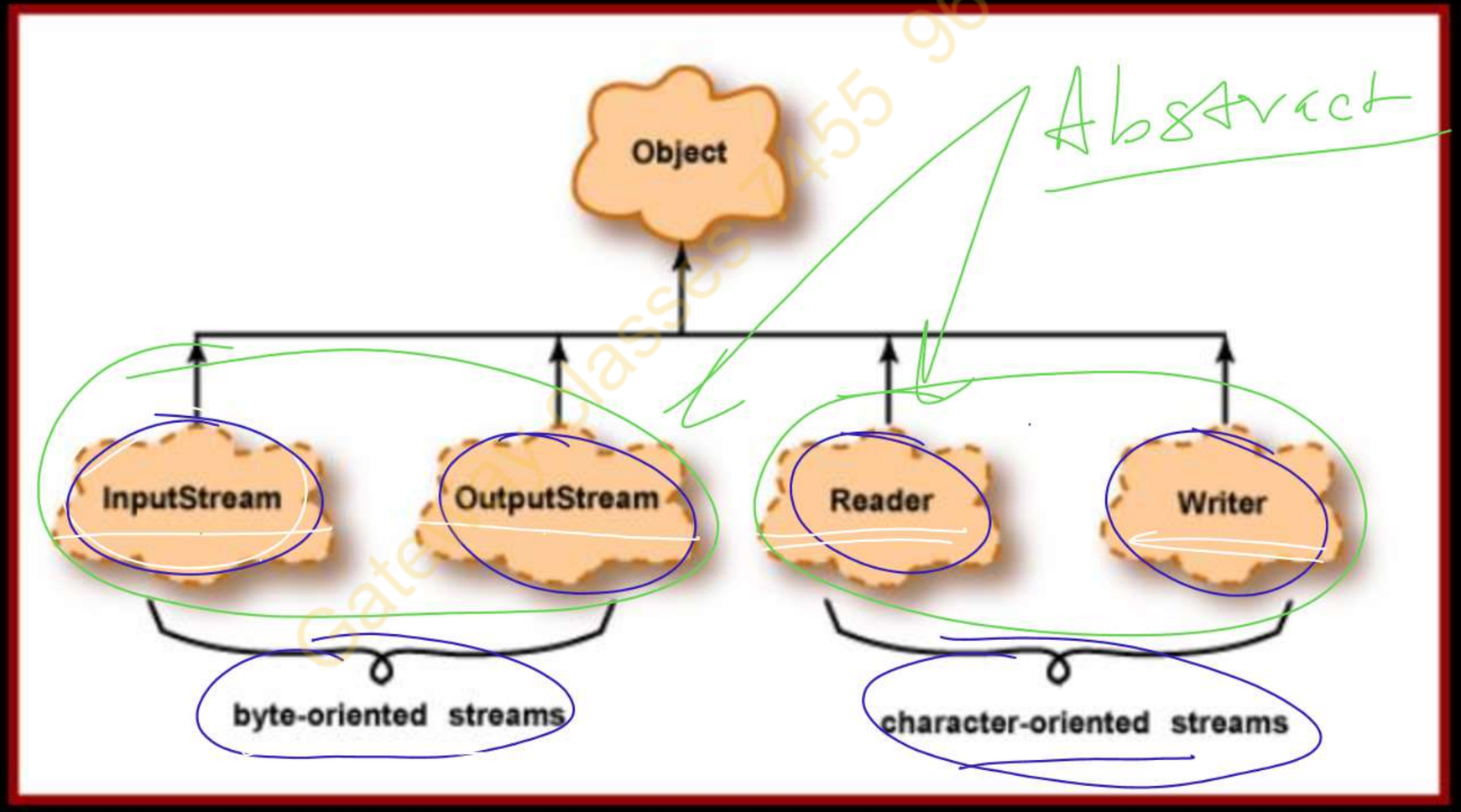
```
            System.out.print("Enter Age:");
            age=sc.nextInt();
            if(age<18 || age>50)
            {
                  System.out.println("Invalid Age");
                  InvalidAgeException ee=new InvalidAgeException();
                  throw ee;
            }
            System.out.println("A1");
            System.out.println("A2");
            System.out.println("A3");
            System.out.println("A4");
      }
}
```

OOPs with JAVA

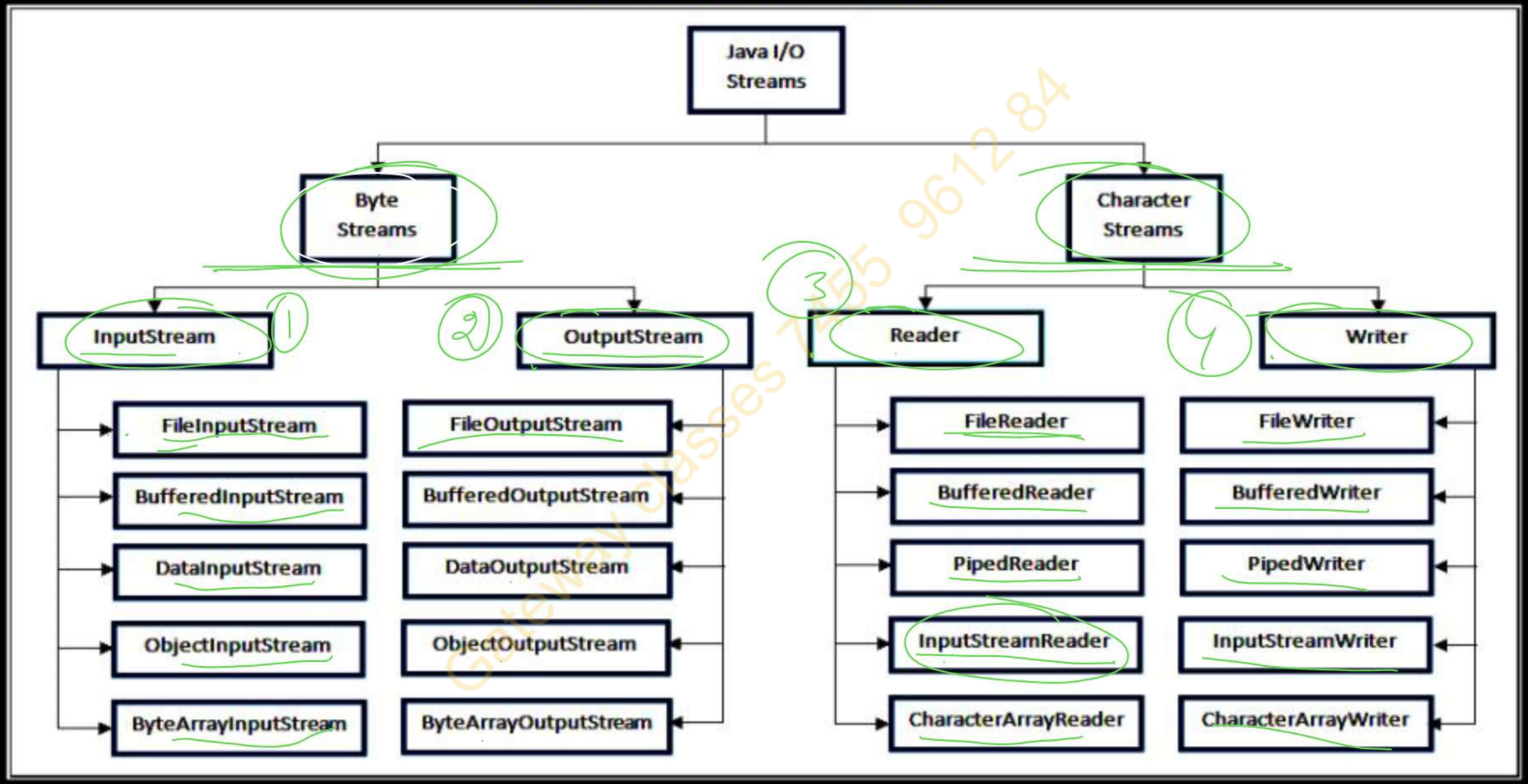
What We will Discuss Today?

- What is IO Package?
- Structure of IO Package?
- Use of IO Classes
- Types of IO Classes
- Difference Between Binary and Text Library
- Reading and Writing Operations in Files
- Examples

IO Package Structure



IO Package Structure

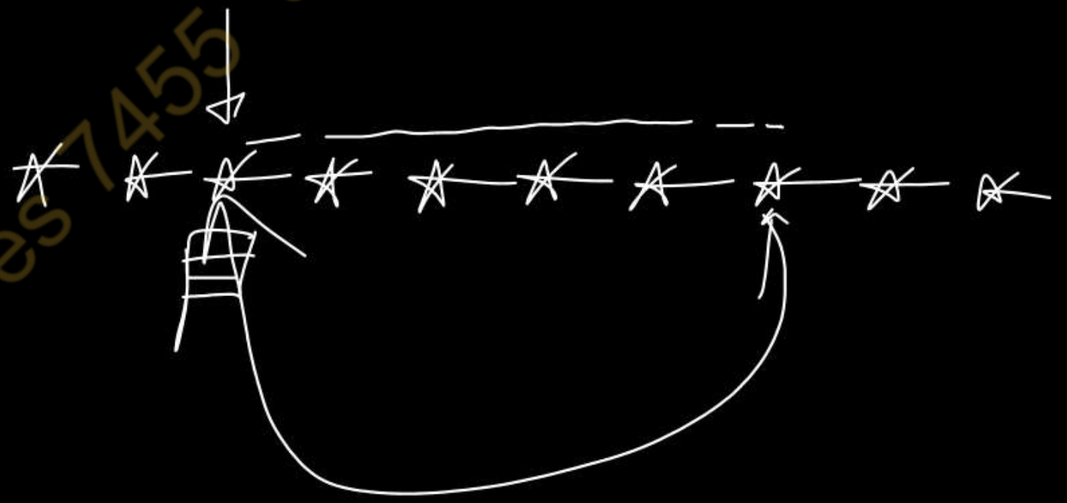


Important Classes

- InputStream
- OutputStream
- Reader
- Writer
- File

Members of InputStream Class

- int read()
- int read(byte[] a)
- int read(byte[] a,int index,int size)
- int skip(int count)
- int available()
- int mark()
- boolean markSupported()
- reset()
- close()



Members of OutputStream Class

- void write(int a)
- int write(byte[] a)
- int write(byte[] a,int index,int size)
- flush()
- close()

byte[] b = {- - - -}

write(b)

Members of Reader Class

- int read()
- int read(char[] a)
- int read(char[] a,int index,int size)
- int skip(int count)
- int mark()
- boolean markSupported()
- reset()
- flush()
- close()

Members of Writer Class

- void write(int a)
- int write(char[] a)
- int write(char[] a,int index,int size)
- flush()
- close()
- write(String s)

Examples

IO Package Programs

```
import java.io.*;
class A47
{
	public static void main(String[] ar)
	{
		try{
			FileInputStream fin=new FileInputStream("d:/a1.txt");
			int a1=fin.read();
			int a2=fin.read();
			int a3=fin.read();
			fin.skip(5);
			byte[] b=new byte[4];
			fin.read(b);
			System.out.println(a1);
			System.out.println(a2);
			System.out.println(a3);
			System.out.println(b[0]);
			System.out.println(b[1]);
			System.out.println(b[2]);
			System.out.println(b[3]);
			int a4=fin.available();
			System.out.println("Remaining:"+a4);
			fin.close();
		}catch(Exception ee)
		{
			System.out.println(ee);
		}
	}
}
```

```
import java.io.*;
class A48
{
	public static void main(String[] ar)
	{
		try{
			FileInputStream fin=new FileInputStream("A2.java");
			int a1=fin.read();
			while(a1>=0)
```

```
                {
                        System.out.print((char)a1);
                        a1=fin.read();
                }
                fin.close();
        }catch(Exception ee)
        {
                System.out.println(ee);
        }
    }
}
```

```
import java.io.*;
class A49
{
    public static void main(String[] ar)
    {
        try{
            FileInputStream fin=new FileInputStream("A2.java");
            int size=fin.available();
            byte[] data=new byte[size];
            fin.read(data);
            String str=new String(data);
            System.out.println(str);
            fin.close();
        }catch(Exception ee)
        {
            System.out.println(ee);
        }
    }
}
```

```
import java.io.*;
class A50
{
    public static void main(String[] ar)
    {
        try{
```

```
                FileOutputStream fout=new FileOutputStream("p1.txt");
                fout.write(65);
                fout.write(66);
                fout.write(67);
                byte[] b={97,98,99,100,101,102,103};
                fout.write(b);
                fout.write(b,2,3);
                fout.close();
            }catch(Exception ee)
            {
                System.out.println(ee);
            }
        }
}
```

```
import java.io.*;
import java.util.*;
class A51
{
    public static void main(String[] ar)
    {
        try{
                Scanner sc=new Scanner(System.in);
                System.out.print("Source File:");
                String src=sc.nextLine();
                System.out.print("Destination File:");
                String dstn=sc.nextLine();
                FileInputStream fin=new FileInputStream(src);
                int size=fin.available();
                byte[] data=new byte[size];
                fin.read(data);
                fin.close();
                FileOutputStream fout=new FileOutputStream(dstn);
                fout.write(data);
                fout.close();
                System.out.println("Done");
        }catch(Exception ee)
        {
            System.out.println(ee);
```

```
		}
	}
}
```

```
import java.io.*;
import java.util.*;
class A52
{
	public static void main(String[] ar)
	{
		try{
				FileReader fr=new FileReader("a5.java");
				BufferedReader bfr=new BufferedReader(fr);
				String s1=bfr.readLine();
				String s2=bfr.readLine();
				String s3=bfr.readLine();
				System.out.println(s1);
				System.out.println(s2);
				System.out.println(s3);
				fr.close();
		}catch(Exception ee)
		{
			System.out.println(ee);
		}
	}
}
```

```
import java.io.*;
import java.util.*;
class A53
{
	public static void main(String[] ar)
	{
		try{
				FileInputStream fr=new FileInputStream("a5.java");
				InputStreamReader inr=new InputStreamReader(fr);
				BufferedReader bfr=new BufferedReader(inr);
				String s1=bfr.readLine();
```

```
                        String s2=bfr.readLine();
                        String s3=bfr.readLine();
                        System.out.println(s1);
                        System.out.println(s2);
                        System.out.println(s3);
                        fr.close();
                }catch(Exception ee)
                {
                        System.out.println(ee);
                }
        }
}
```

```
import java.io.*;
import java.util.*;
class A54
{
        public static void main(String[] ar)
        {
                try{
                        InputStreamReader inr=new InputStreamReader(System.in);
                        BufferedReader bfr=new BufferedReader(inr);
                        String s1=bfr.readLine();
                        String s2=bfr.readLine();
                        String s3=bfr.readLine();
                        System.out.println(s1);
                        System.out.println(s2);
                        System.out.println(s3);
                }catch(Exception ee)
                {
                        System.out.println(ee);
                }
        }
}
```

IO Exercise

Write a Java program to get a list of all file/directory names in the given directory.

(Hint: Use File Class)

Write a Java program to check if a file or directory specified by pathname exists or not.

(Hint: Use File Class)

Write a Java program to read input from the Java console.

Write a Java program to read file content line by line.

Write a Java program to store text file content line by line in an array.

Write a Java program to read the first 3 lines of a file.

Write a Java program to find the longest word in a text file.

OOPs with JAVA

What We will Discuss Today?

- What is Thread?
- Why we need Threads?
- Threads v/s Process
- Threads in Java
- Thread Life Cycle
- Thread Priorities
- Thread Synchronization
- Inter Thread Communication

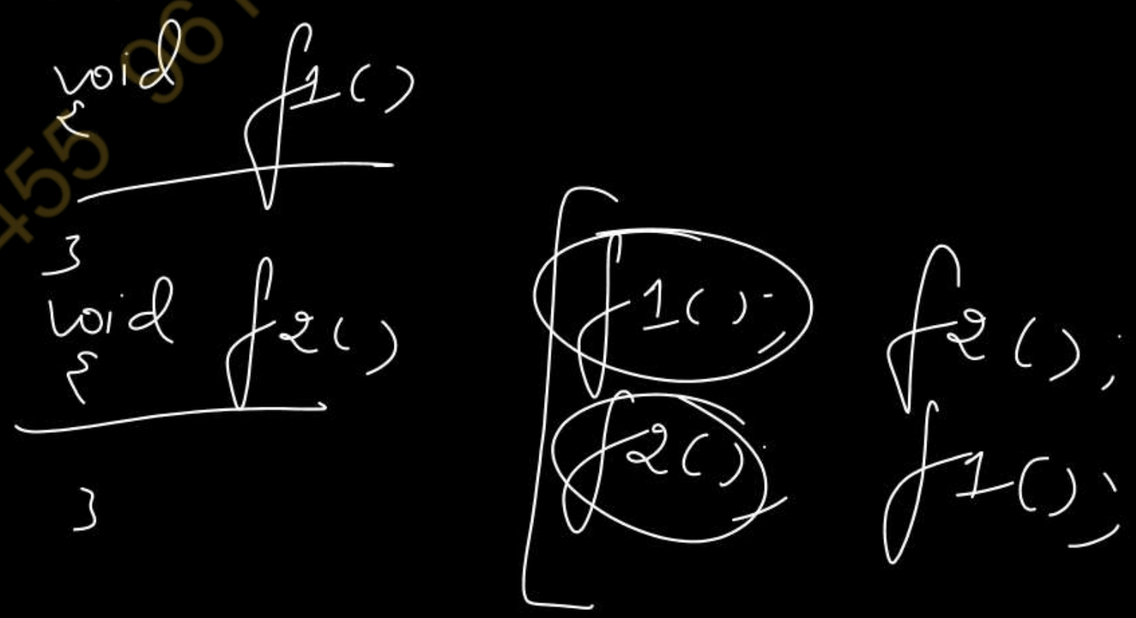
What is Thread?

A Thread is light weight Process with a Thread ID, Program Counter, Register Set, Stack etc. as same in Process. But Thread shares code, data etc. with the other threads in the Same Process.

Thread is a Basic Unit of CPU Execution.

We can use thread to implement Parallelism.

*Threads are very Important for Interview



Thread v/s Process

Parameter	Process	Thread
Definition	Process means a program is in execution.	Thread means a segment of a process.
Lightweight	The process is not Lightweight.	Threads are Lightweight.
Termination time	The process takes more time to terminate.	The thread takes less time to terminate.
Creation time	It takes more time for creation.	It takes less time for creation.
Communication	Communication between processes needs more time compared to thread.	Communication between threads requires less time compared to processes.
Context switching time	It takes more time for context switching.	It takes less time for context switching.
Resource	Process consume more resources.	Thread consume fewer resources.
Treatment by OS	Different process are tread separately by OS.	All the level peer threads are treated as a single task by OS.
Memory	The process is mostly isolated.	Threads share memory.
Sharing	It does not share data	Threads share data with each other.

Threads in Java

We can create thread by extending “Thread” Class or by the Implementation of “Runnable” Interface.

```
class MyThread extends Thread
{
        public void run()
        {
                //----------
                //----------
        }
}
```

```
class MyThread implements Runnable
{
        public void run()
        {
                //----------
                //----------
        }
}
```

Thread Members (Constructors and Functions)

Thread t = new Thread();

Constructors

- Thread()
- Thread(String name)
- Thread(Runnable target)
- Thread(Runnable target, String name)

Functions

- void start()
- void run()
- void stop()
- void suspend()
- boolean isAlive()
- void setName(String name)
- String getName()
- void setPriority(int priority)
- int getPriority()

void resume()

Static Members

- Thread currentThread()
- void sleep(int millisecond)
- void sleep(int millisecond,int nanosecond)

Thread Life Cycle

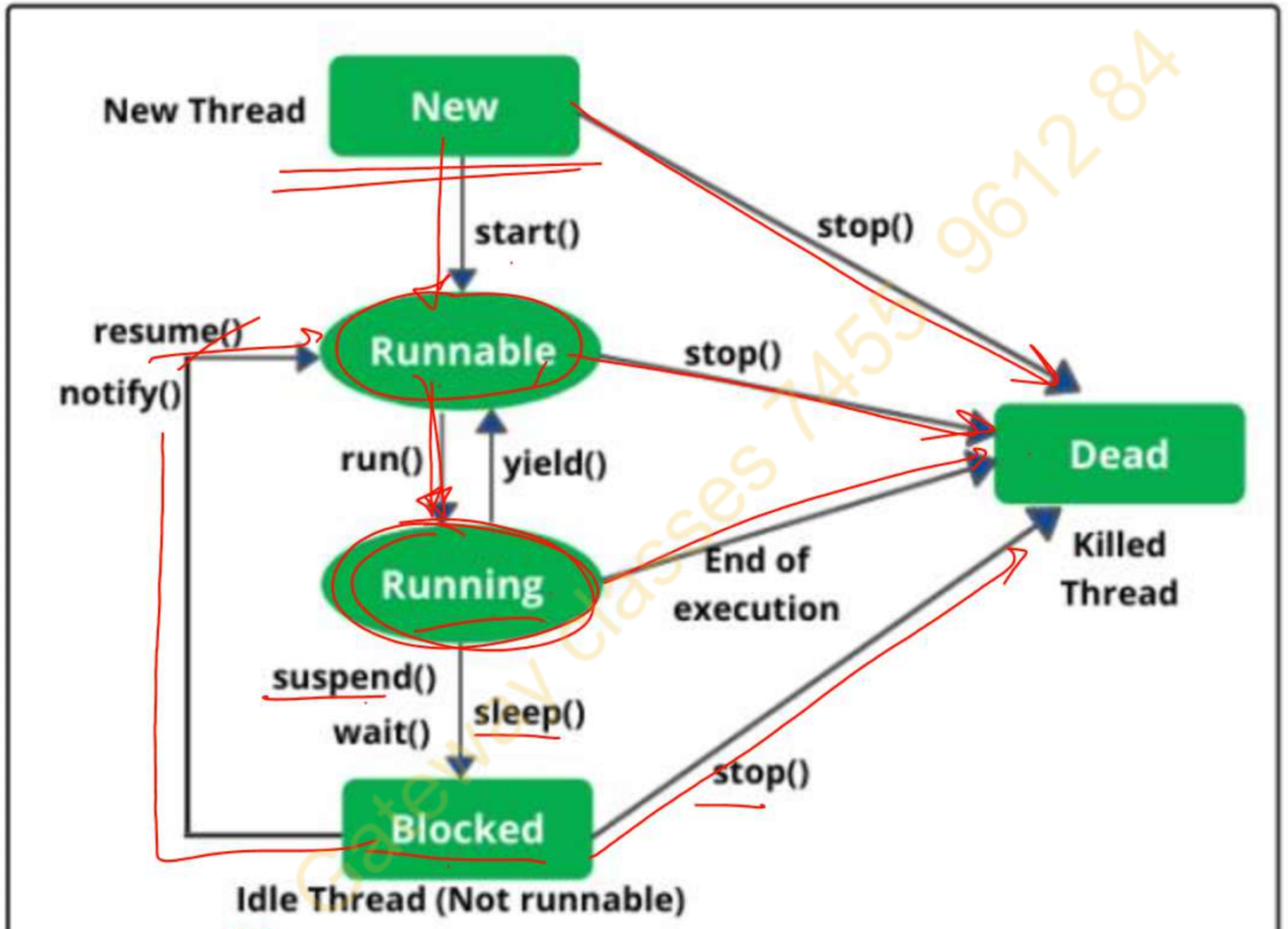


Fig: State Transition Diagram of a Thread

Threads Programs

```
class Thread1 extends Thread
{
	public void run()
	{
		for(int i=1;i<=10;i++)
		{
			System.out.println("I="+i);
		}
	}
}
class Thread2 implements Runnable
{
	public void run()
	{
		for(int k=11;k<=20;k++)
		{
			System.out.println("K="+k);
		}
	}
}
class A55
{
	public static void main(String[] ar)
	{
		Thread1 t1=new Thread1();
		Thread t2=new Thread(new Thread2());
		System.out.println("Starting 1st");
		t1.start();
		System.out.println("Starting 2nd");
		t2.start();
		System.out.println("End of Main");
	}
}
```

```
class A56
{
```

```
	public static void main(String[] ar)
	{
		for(int i=1;i<=10;i++)
		{
			System.out.println(i);
			try{
				Thread.sleep(1000);
			}catch(Exception ee){}
		}
	}
}
```

```
class Shared
{
	int x=10;
}
class Thread1 extends Thread
{
	Shared ss;
	Thread1(Shared s)
	{
		ss=s;
	}
	public void run()
	{
		System.out.println("A1");
		System.out.println("A2");
		System.out.println("A3");
		// Critical Section Start
		synchronized(ss)
		{
			System.out.print("{");
			System.out.print("X="+ss.x);
			ss.x=ss.x*2;
			System.out.print(" X="+ss.x);
			System.out.println("}");
		}
		// Critical End
		System.out.println("A4");
```

```
            System.out.println("A5");
            System.out.println("A6");
        }
}
class Thread2 extends Thread
{
        Shared ss;
        Thread2(Shared s)
        {
            ss=s;
        }
        public void run()
        {
            System.out.println("B1");
            System.out.println("B2");
            System.out.println("B3");
            // Critical Section Start
            synchronized(ss)
            {
                System.out.print("[");
                System.out.print("x="+ss.x);
                ss.x=ss.x+2;
                System.out.print(" x="+ss.x);
                System.out.println("]");
            }
            // Critical Section End
            System.out.println("B4");
            System.out.println("B5");
            System.out.println("B6");
        }
}
class A57
{
        public static void main(String[] ar)
        {
            Shared s=new Shared();
            Thread1 t1=new Thread1(s);
            Thread2 t2=new Thread2(s);
            t1.start();
            t2.start();
```

```
	}
}
-----------------------------------------------------------------

class Customers
{
	int amount=0;
	synchronized void deposit(int amt)
	{
		System.out.println("Initial:"+amount);
		amount=amount+amt;
		System.out.println("After Deposit:"+amount);
		notify();
	}
	synchronized void withdrawl(int amt)
	{
		try{
			wait();
		}catch(Exception ee){}
		System.out.println("Initial:"+amount);
		amount=amount-amt;
		System.out.println("After Withdraw:"+amount);
	}
}
class Thread1 extends Thread
{
	Customers cc;
	Thread1(Customers c)
	{
		cc=c;
	}
	public void run()
	{
		cc.withdrawl(2000);
	}
}
class Thread2 extends Thread
{
	Customers cc;
	Thread2(Customers c)
```

```
	{
		cc=c;
	}
	public void run()
	{
		cc.deposit(5000);
	}
}
class A58
{
	public static void main(String[] ar)
	{
		Customers c=new Customers();
		Thread1 t1=new Thread1(c);
		Thread2 t2=new Thread2(c);
		t1.start();
		t2.start();
	}
}
```

Thread Exercise

Write a Java program that sorts an array of integers using multiple threads.

Write a Java program that calculates the sum of all prime numbers up to a given limit using multiple threads.

Write a Java program that creates a bank account with concurrent deposits and withdrawals using threads.

Write a Java program to demonstrate Semaphore usage for thread synchronization

AKTU

B. Tech II-Year

Gateway Classes

Full Courses Available in App

AKTU B.Tech I- Year : All Branches

AKTU B.Tech II- Year

Branches : 1. CS IT & Allied

2. EC & Allied

3. ME & Allied

4. EE & Allied

Downlod App Now

Download App

Full Courses

- V. Lectures
- Pdf Notes
- AKTU PYQs
- DPP