OBJECT ORIENTED TECHNIQUES USING JAVA(ACSE0302)

Unit: 4

Concurrency in Java and I/O Stream

Course Details
(B.Tech 3rd Sem /2nd Year)

Dr. Ratna Nitin Patil
Department Artificial Intelligence

Table of Contents

Conti….

Conti…

Brief Introduction

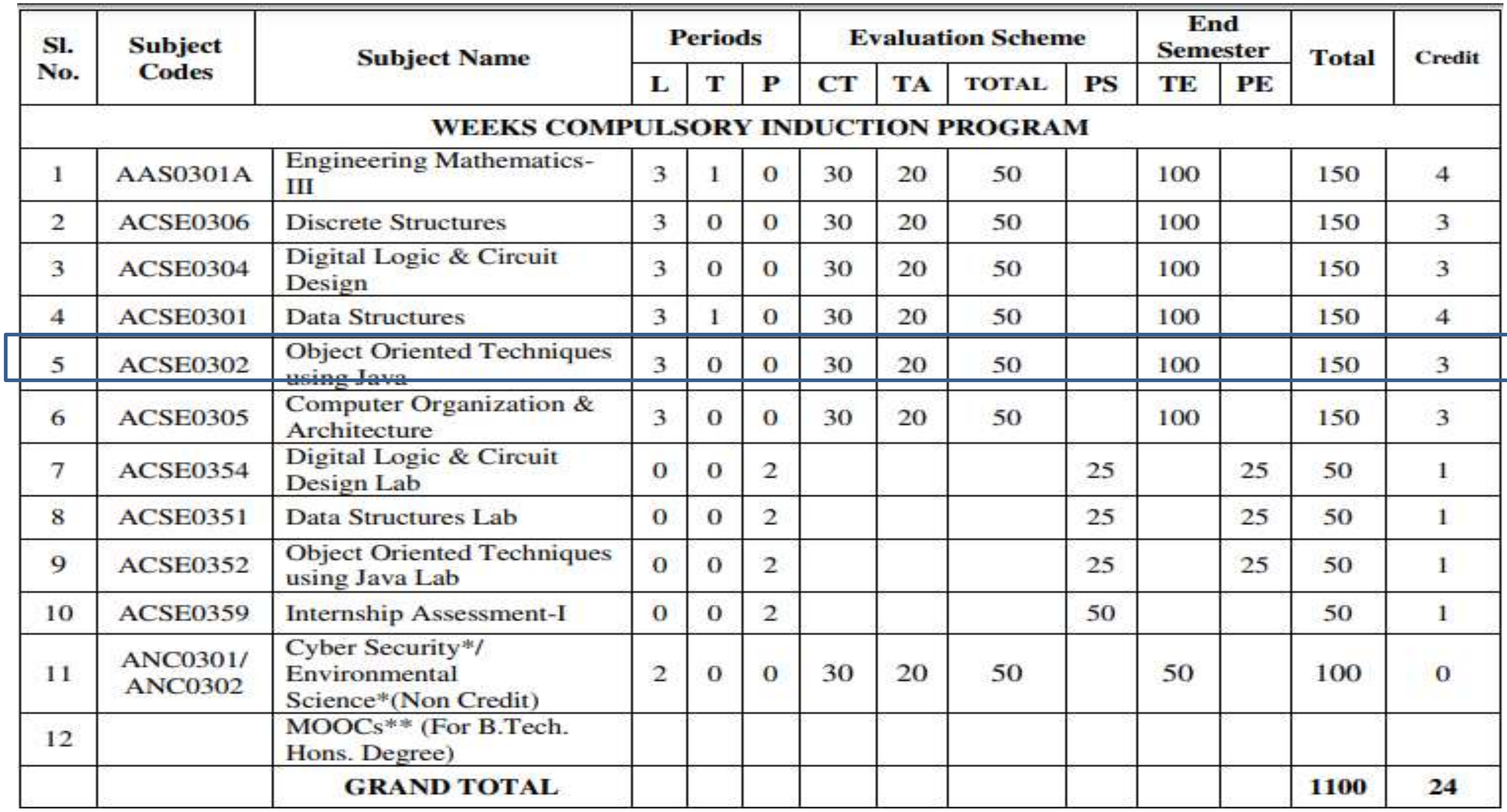
I hold Bachelor of Engineering degree from University of Pune, Maharashtra and completed Masters in Computer Engineering from Thapar University, Patiala, Punjab in 2001. I completed PhD in Computer Engineering from Dr. Babasaheb Ambedkar Marathwada University, Aurangabad in 2021.

I worked as an Associate Professor at Kanpur Institute of Technology, Kanpur, St Peters College of Engineering, Chennai, Sriram College of Engineering, Chennai, Mar Baselios College of Engineering, Trivandrum.

I also worked at VIT, Pune & Presidency University, Bangalore. I have approximately 25 years of teaching experience at various Institutes / Engineering Institutions in different parts of the country.

My research interest includes Machine Learning, Natural Language Processing and Analysis of Algorithms.

Evaluation Scheme



Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
WEEKS COMPULSORY INDUCTION PROGRAM													
1	AAS0301A	Engineering Mathematics-III	3	1	0	30	20	50		100		150	4
2	ACSE0306	Discrete Structures	3	0	0	30	20	50		100		150	3
3	ACSE0304	Digital Logic & Circuit Design	3	0	0	30	20	50		100		150	3
4	ACSE0301	Data Structures	3	1	0	30	20	50		100		150	4
5	ACSE0302	Object Oriented Techniques using Java	3	0	0	30	20	50		100		150	3
6	ACSE0305	Computer Organization & Architecture	3	0	0	30	20	50		100		150	3
7	ACSE0354	Digital Logic & Circuit Design Lab	0	0	2				25		25	50	1
8	ACSE0351	Data Structures Lab	0	0	2				25		25	50	1
9	ACSE0352	Object Oriented Techniques using Java Lab	0	0	2				25		25	50	1
10	ACSE0359	Internship Assessment-I	0	0	2				50			50	1
11	ANC0301/ ANC0302	Cyber Security*/ Environmental Science*(Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs** (For B.Tech. Hons. Degree)											
		GRAND TOTAL										1100	24

Syllabus

UNIT-I	Introduction	8 Hours
Object Oriented Programming: Introduction and Features: Abstraction, Encapsulation, Polymorphism, and Inheritance. Modeling Concepts: Introduction, Class Diagram and Object Diagram. Control Statements: Decision Making, Looping and Branching, Argument Passing Mechanism: Command Line Argument		
UNIT-II	**Basics of Java Programming**	8 Hours
Class and Object: Object Reference, Constructor, Abstract Class, Interface and its uses, Defining Methods, Use of "this" and "super" keyword, Garbage Collection and finalize () Method. Inheritance: Introduction and Types of Inheritance in Java, Constructors in Inheritance. Polymorphism: Introduction and Types, Overloading and Overriding. Lambda expression: Introduction and Working with Lambda Variables. Arrays: Introduction and its Types.		

Syllabus

UNIT-III	Packages, Exception Handling and String Handling	8 hours
Packages: Introduction and Types, Access Protection in Packages, Import and Execution of Packages. Exception Handling, Assertions and Localizations: Introduction and Types, Exceptions vs. Errors, Handling of Exception. Finally, Throws and Throw keyword, Multiple Catch Block, Nested Try and Finally Block. Assertions and Localizations Concepts and its working. String Handling: Introduction and Types, Operations, Immutable String, Method of String class, String Buffer and String Builder class		

UNIT-IV	Concurrency in Java and I/O Stream	8 hours
Threads: Introduction and Types, Creating Threads, Thread Life-Cycle, Thread Priorities, Daemon Thread, Runnable Class, Synchronizing Threads. I/O Stream: Introduction and Types, Common I/O Stream Operations, Interaction with I/O Streams Classes. Annotations: Introduction, Custom Annotations and Applying Annotations.		
UNIT-V	**GUI Programming, Generics and Collections**	8 hours
GUI Programming: Introduction and Types, Applet, Life Cycle of Applet, AWT, Components and Containers, Layout Managers and User-Defined Layout and Event Handling. Generics and Collections: Introduction, Using Method References, Using Wrapper Class, Using Lists, Sets, Maps and Queues, Working with Generics.		

Branch-wise Applications

Java can be used :

- Data import and export.
- Cleaning data.
- Statistical analysis.
- Machine learning and Deep learning.
- Deep learning.
- Text analytics (also known as Natural Language Processing or NLP).
- Data visualization.

Course Objectives

- The objective of this course is to understand the object-oriented methodology and its techniques to design and develop conceptual models and demonstrate the standard concepts of object-oriented techniques modularity, I/O. and other standard language constructs.

- The basic objective of this course is to understand the fundamental concepts of object-oriented programming in Java language and also implement the Multithreading concepts, GUI based application and collection framework.

Course Outcomes

After completion of this course students will be able to:

CO1	**Identify the concepts of object oriented programming and relationships among them needed in modeling.**
CO2	**Demonstrate the Java programs using OOP principles and also implement the concepts of lambda expressions.**
CO3	**Implement packages with different protection level resolving namespace collision and evaluate the error handling concepts for uninterrupted execution of Java program.**
CO4	**Implement Concurrency control, I/O Streams and Annotations concepts by using Java program.**
CO5	Design and develop the GUI based application, Generics and Collections in Java programming language to solve the real-world problem.

Program Outcomes

1. Engineering knowledge:
2. Problem analysis:
3. Design/development of solutions:
4.Conduct investigations of complex problems
5. Modern tool usage
6. The engineer and society
7. Environment and sustainability
8. Ethics:
9. Individual and team work
10. Communication:
11. Project management and finance
12. Life-long learning

NIET Greater Noida
GET FUTURE READY
Estd 2001
AN AUTONOMOUS INSTITUTE

CO-PO Mapping

Mapping of Course Outcomes and Program Outcomes:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO.1	**3**	**3**	**3**	**2**	**1**				**2**			**3**
CO.2	3	3	3	2	1				2			3
CO.3	3	3	3	2	1				2			3
CO.4	3	3	3	2	1				2			3
CO.5	3	3	3	2	1				2			3

Program Specific Outcomes

On successful completion of graduation degree the Engineering graduates will be able to:

PSO1: The ability to identify, analyze real world problems and design their ethical solutions using artificial intelligence, robotics, virtual/augmented reality, data analytics, block chain technology, and cloud computing.

PSO2: The ability to design and develop the hardware sensor devices and related interfacing software systems for solving complex engineering problems.

PSO3: The ability to understand inter disciplinary computing techniques and to apply them in the design of advanced computing.

PSO4: The ability to conduct investigation of complex problem with the help of technical, managerial, leadership qualities, and moder engineering tools provided by industry sponsored laboratories.

CO-PSO Mapping

Mapping of Course Outcomes and Program Specific Outcomes:

	PSO1	PSO2	PSO3	PSO4
Co.1	**2**	**3**	**2**	**1**
CO.2	**2**	**3**	**2**	**1**
CO.3	2	3	2	1
CO.4	**2**	**3**	**2**	**1**
CO.5	2	3	2	1

Program Educational Objectives

PEO1: To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.

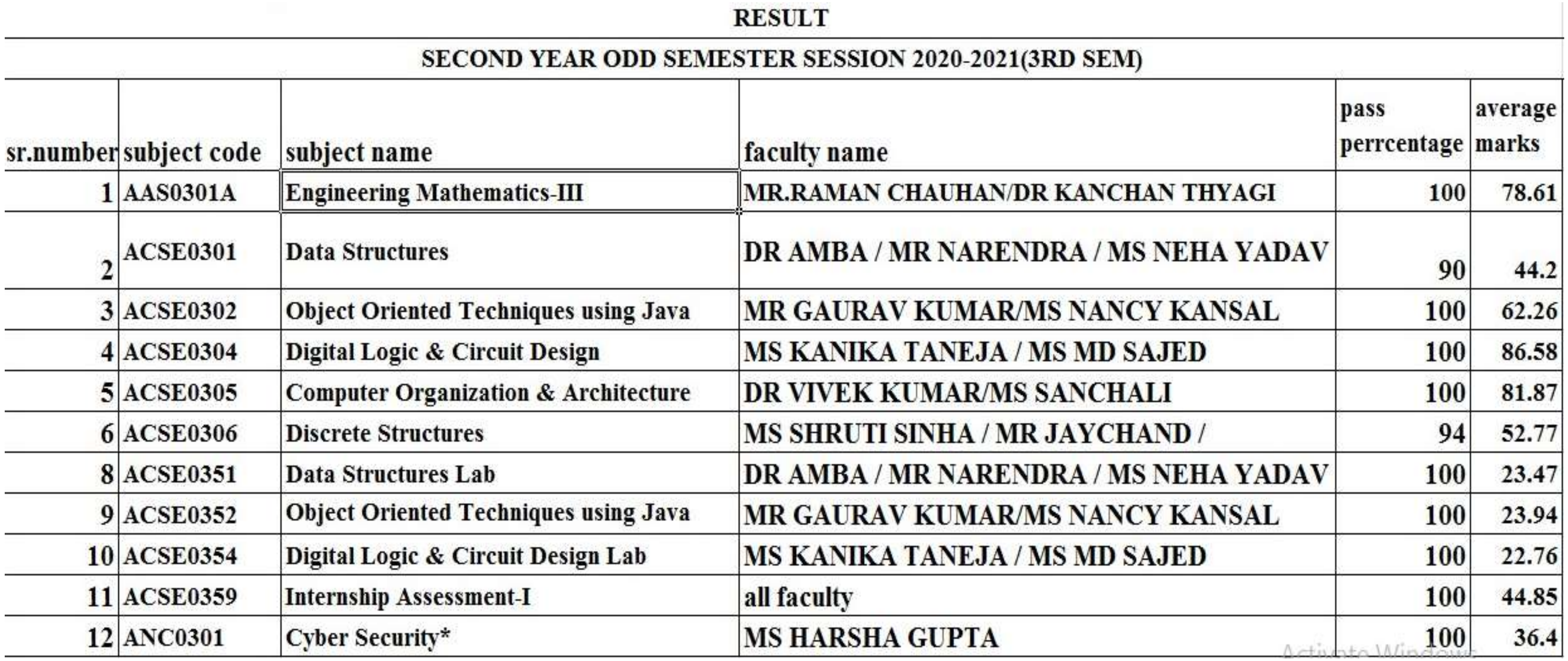
PEO2: To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face global challenges.

PEO3: To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.

PEO4: To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, entrepreneur and bureaucrat for betterment of society

NIET Greater Noida GET FUTURE READY Estd 2001 AN AUTONOMOUS INSTITUTE

Result Analysis

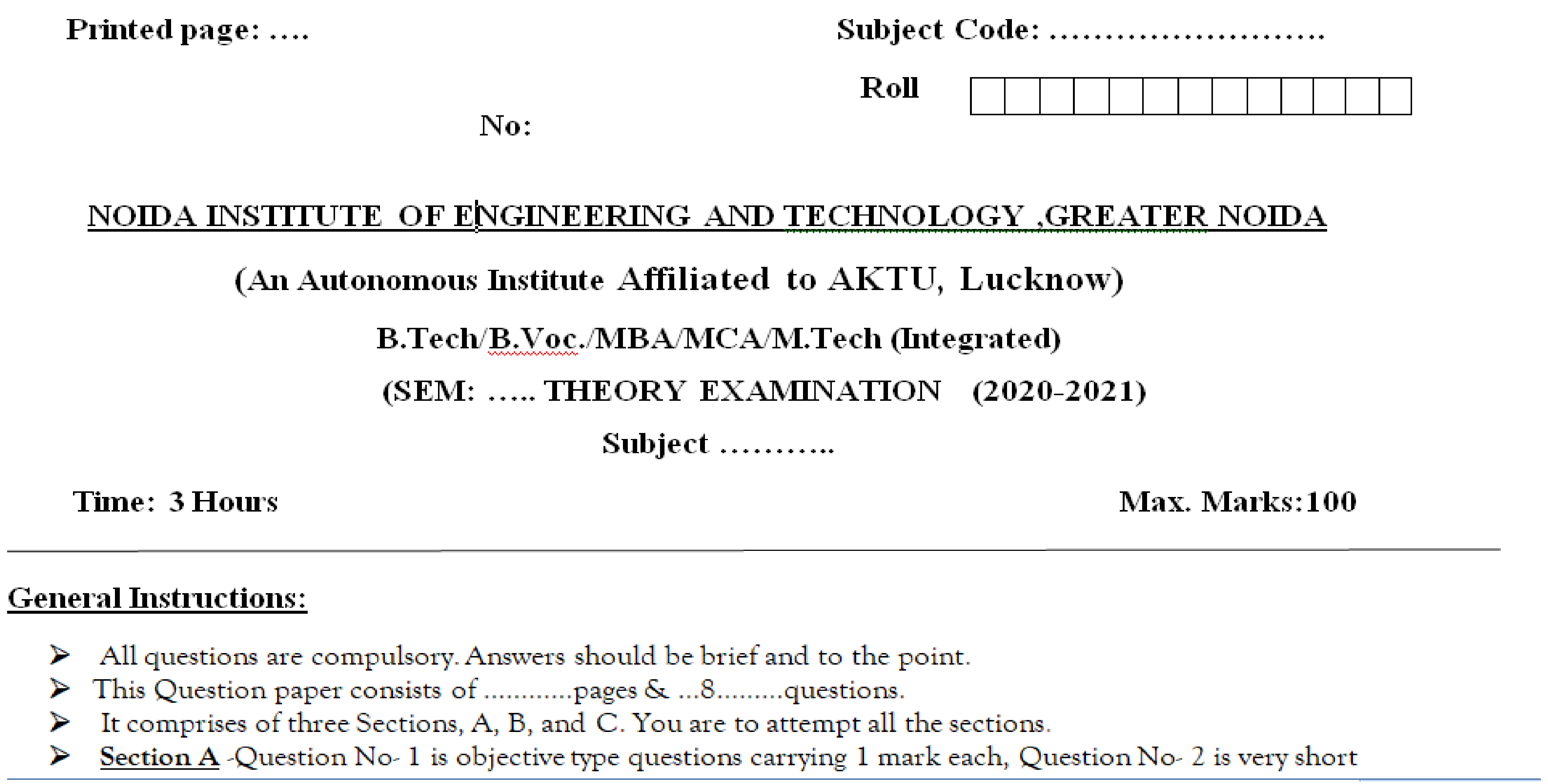


RESULT

SECOND YEAR ODD SEMESTER SESSION 2020-2021(3RD SEM)

sr.number	subject code	subject name	faculty name	pass perrcentage	average marks
1	AAS0301A	Engineering Mathematics-III	MR.RAMAN CHAUHAN/DR KANCHAN THYAGI	100	78.61
2	ACSE0301	Data Structures	DR AMBA / MR NARENDRA / MS NEHA YADAV	90	44.2
3	ACSE0302	Object Oriented Techniques using Java	MR GAURAV KUMAR/MS NANCY KANSAL	100	62.26
4	ACSE0304	Digital Logic & Circuit Design	MS KANIKA TANEJA / MS MD SAJED	100	86.58
5	ACSE0305	Computer Organization & Architecture	DR VIVEK KUMAR/MS SANCHALI	100	81.87
6	ACSE0306	Discrete Structures	MS SHRUTI SINHA / MR JAYCHAND /	94	52.77
8	ACSE0351	Data Structures Lab	DR AMBA / MR NARENDRA / MS NEHA YADAV	100	23.47
9	ACSE0352	Object Oriented Techniques using Java	MR GAURAV KUMAR/MS NANCY KANSAL	100	23.94
10	ACSE0354	Digital Logic & Circuit Design Lab	MS KANIKA TANEJA / MS MD SAJED	100	22.76
11	ACSE0359	Internship Assessment-I	all faculty	100	44.85
12	ANC0301	Cyber Security*	MS HARSHA GUPTA	100	36.4

End Semester Question Paper Templates (Offline Pattern/Online Pattern



Printed page: ….   Subject Code: ……………………..

Roll No:

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY ,GREATER NOIDA

(An Autonomous Institute Affiliated to AKTU, Lucknow)

B.Tech/B.Voc./MBA/MCA/M.Tech (Integrated)

(SEM: ….. THEORY EXAMINATION (2020-2021)

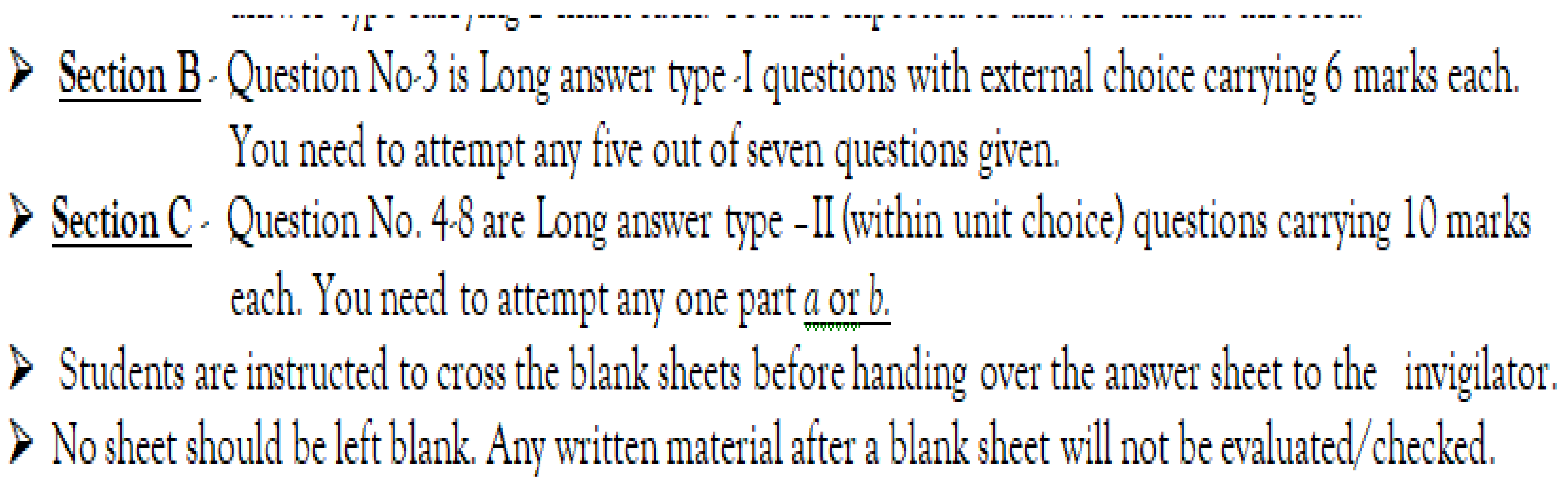
Subject ………..

Time: 3 Hours   Max. Marks:100

General Instructions:

- ➢ All questions are compulsory. Answers should be brief and to the point.
- ➢ This Question paper consists of ………..pages & …8………questions.
- ➢ It comprises of three Sections, A, B, and C. You are to attempt all the sections.
- ➢ Section A -Question No- 1 is objective type questions carrying 1 mark each, Question No- 2 is very short

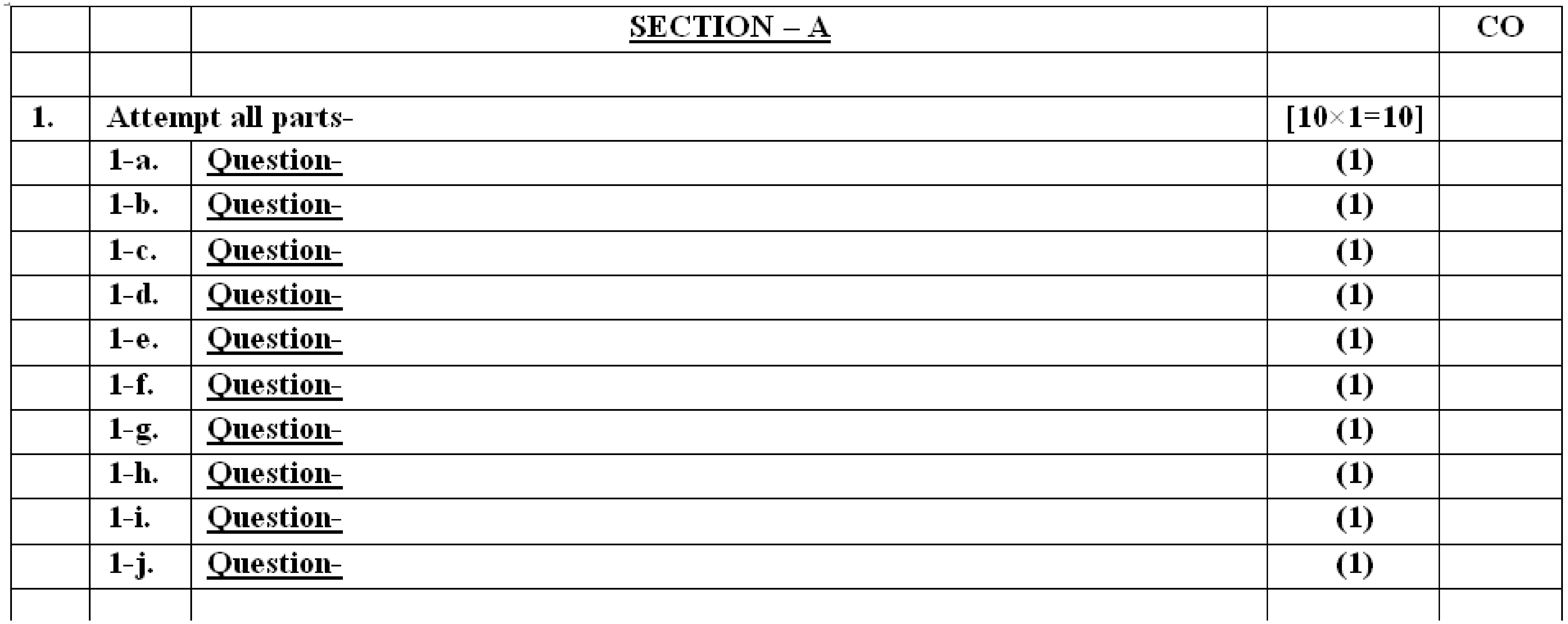
End Semester Question Paper Templates (Offline Pattern/Online Pattern



[illegible]

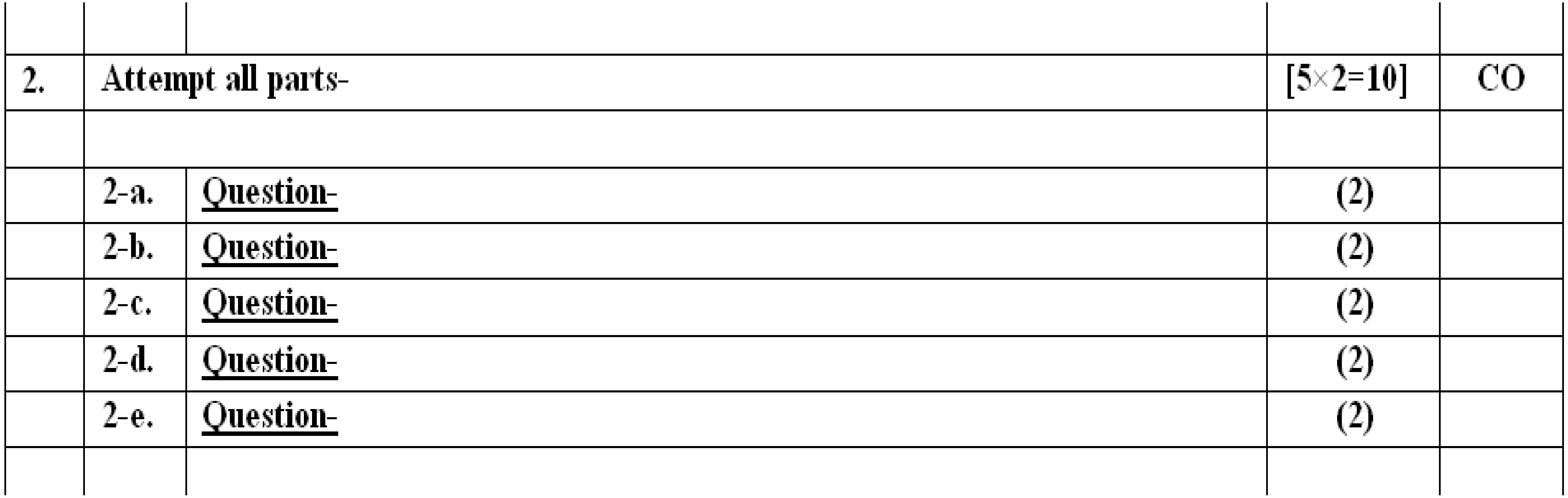
- **Section B** - Question No-3 is Long answer type -I questions with external choice carrying 6 marks each. You need to attempt any five out of seven questions given.
- **Section C** - Question No. 4-8 are Long answer type –II (within unit choice) questions carrying 10 marks each. You need to attempt any one part a or b.
- Students are instructed to cross the blank sheets before handing over the answer sheet to the invigilator.
- No sheet should be left blank. Any written material after a blank sheet will not be evaluated/checked.

End Semester Question Paper Templates (Offline Pattern/Online Pattern



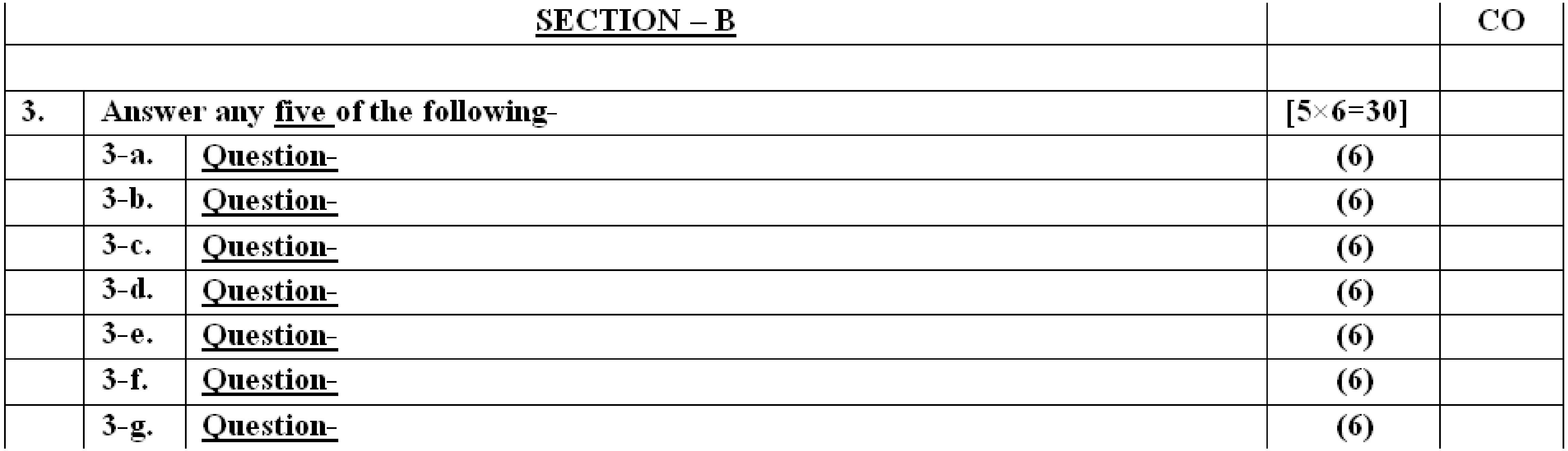
		SECTION – A		**CO**
1.	**Attempt all parts-**		**[10×1=10]**	
	1-a.	**Question-**	**(1)**	
	1-b.	**Question-**	**(1)**	
	1-c.	**Question-**	**(1)**	
	1-d.	**Question-**	**(1)**	
	1-e.	**Question-**	**(1)**	
	1-f.	**Question-**	**(1)**	
	1-g.	**Question-**	**(1)**	
	1-h.	**Question-**	**(1)**	
	1-i.	**Question-**	**(1)**	
	1-j.	**Question-**	**(1)**	

End Semester Question Paper Templates (Offline Pattern/Online Pattern



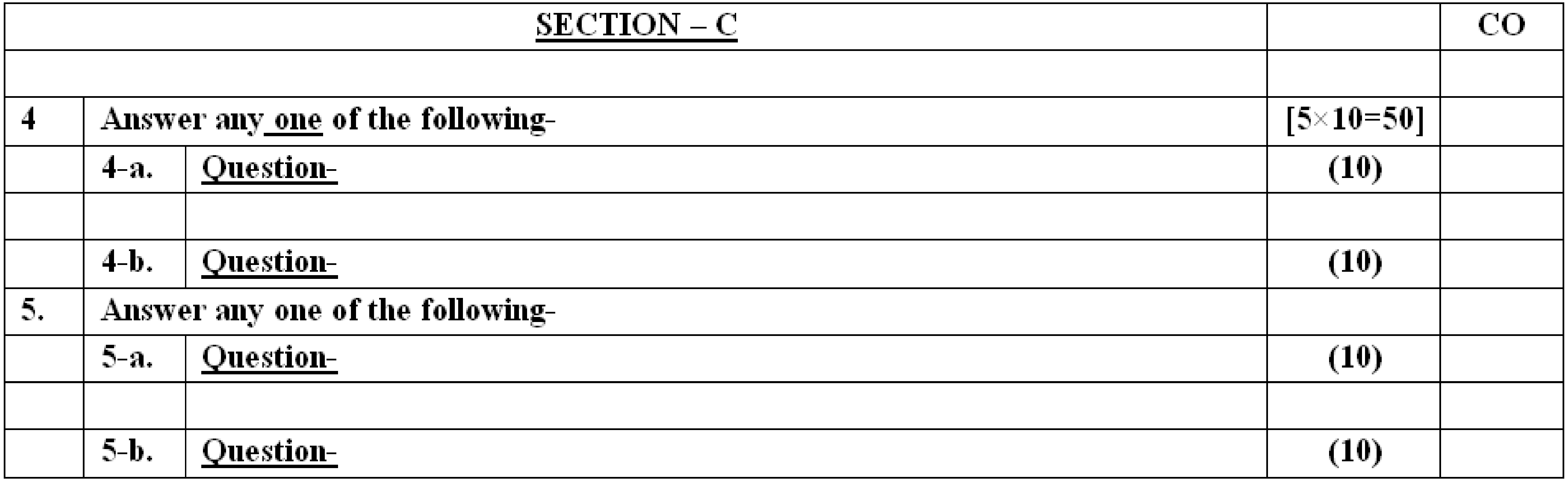
2.	Attempt all parts-		[5×2=10]	CO
	2-a.	**<u>Question-</u>**	(2)	
	2-b.	**<u>Question-</u>**	(2)	
	2-c.	**<u>Question-</u>**	(2)	
	2-d.	**<u>Question-</u>**	(2)	
	2-e.	**<u>Question-</u>**	(2)	

End Semester Question Paper Templates (Offline Pattern/Online Pattern



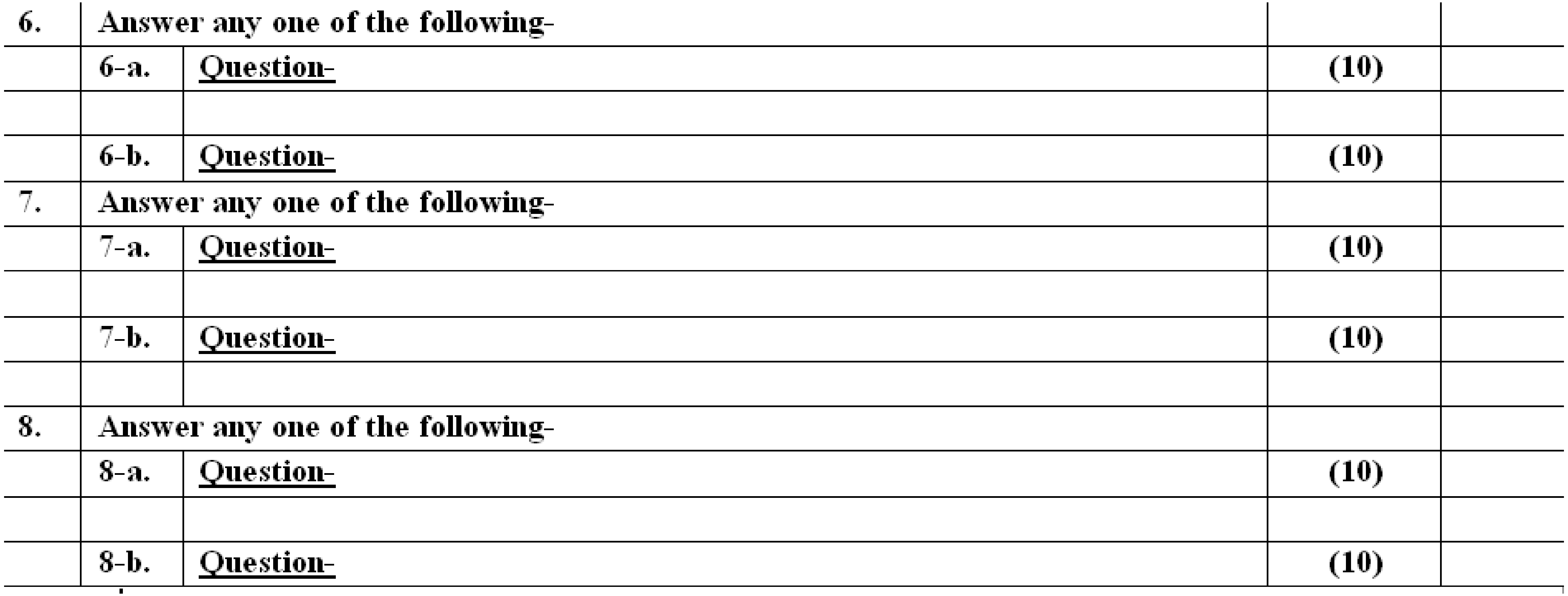
		SECTION – B		CO
3.	**Answer any five of the following-**		**[5×6=30]**	
	3-a.	**Question-**	**(6)**	
	3-b.	**Question-**	**(6)**	
	3-c.	**Question-**	**(6)**	
	3-d.	**Question-**	**(6)**	
	3-e.	**Question-**	**(6)**	
	3-f.	**Question-**	**(6)**	
	3-g.	**Question-**	**(6)**	

End Semester Question Paper Templates (Offline Pattern/Online Pattern



SECTION – C				**CO**
4	**Answer any one of the following-**		**[5×10=50]**	
	4-a.	**Question-**	**(10)**	
	4-b.	**Question-**	**(10)**	
5.	**Answer any one of the following-**			
	5-a.	**Question-**	**(10)**	
	5-b.	**Question-**	**(10)**	

End Semester Question Paper Templates (Offline Pattern/Online Pattern



6.	**Answer any one of the following-**			
	6-a.	**<u>Question-</u>**	**(10)**	
	6-b.	**<u>Question-</u>**	**(10)**	
7.	**Answer any one of the following-**			
	7-a.	**<u>Question-</u>**	**(10)**	
	7-b.	**<u>Question-</u>**	**(10)**	
8.	**Answer any one of the following-**			
	8-a.	**<u>Question-</u>**	**(10)**	
	8-b.	**<u>Question-</u>**	**(10)**	

Prerequisite/Recap

- Student must know at least the basics of how to use a computer, and should be able to start a command line shell.
- Knowledge of basic programming concepts, as covered in ‘Programming Basic' course is necessary.
- Students must have basic understanding of computer programming and related programming paradigms

About the Subject with videos

- OOTS refers to languages that uses objects in programming.
- OOTS aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming.
- The main aim of OOTS is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function

Link:	
Unit 1	https://www.youtube.com/watch?v=r59xYe3Vyks&list=PLS1QulWo1RIbfTjQvTdj8Y6yyq4R7g-Al
Unit 2	https://www.youtube.com/watch?v=ZHLdVRXIuC8&list=PLS1QulWo1RIbfTjQvTdj8Y6yyq4R7g-Al&index=18,
Unit 3	https://www.youtube.com/watch?v=hBh_CC5y8-s ____ https://www.youtube.com/watch?v=OjdT2l-EZJA
Unit 4	https://www.youtube.com/watch?v=qQVqfvs3p48
Unit 5	https://www.youtube.com/watch?v=2qWPpgALJyw

UNIT CONTENT

1. Threads: Introduction and Types
   - Creating Threads
   - Thread Life-Cycle
   - Thread Priorities
   - Daemon Thread
   - Runnable Class
   - Synchronizing Threads
2. I/O Stream: Introduction and Types
   - Common I/O Stream Operations
   - Interaction with I/O Streams Classes
3. Annotations: Introduction
   - Custom Annotations
   - Applying Annotations

Unit Objectives

- This unit gives the fundamental knowledge of threads, I/O control and annotations.
- The unit will help the students to understand how the approaches of threads help in solving real life problems.
- Basic understanding of I/O operations used.
- Overview of creating custom annotations and using them in programs.

Topic Objectives

After you have read and studied this topic, you should be able to

- To understand Concurrency in Java.
- To learn the concept of thread, Lifecycle of thread, Types, etc.
- To understand the application of Annotations in Java.
- To explore the knowledge of I/O Stream operations in Java.

Topic mapping with CO

Topic	**CO**
Threads: Introduction and Types	CO4
Creating Threads	CO4
Thread Life-Cycle	CO4
Thread Priorities	CO4
Daemon Thread, Runnable Class	CO4
I/O Stream: Introduction and Types	CO4
Common I/O Stream Operations	CO4
Interaction with I/O Streams Classes	CO4
Annotations: Introduction	CO4
Custom Annotations	CO4
Applying Annotations	CO4

Lecture 1

1. Threads: Introduction and Types
   - Creating Threads
   - Thread Life-Cycle

Threads: Introduction

- Threads
  - Threads allows a program to operate more efficiently by doing multiple things at the same time.
  - Threads can be used to perform complicated tasks in the background without interrupting the main program.
  - Java is a multi-threaded programming language which means we can develop multi-threaded program using Java.
  - A multi-threaded program contains two or more parts that can run concurrently, and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.
  - Multitasking is when multiple processes share common processing resources such as a CPU.

Threads: Introduction (CO4)

- Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads.
- Each of the threads can run in parallel.
- The OS divides processing time not only among different applications, but also among each thread within an application.
- Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program.

Types of Threads (CO4)

- Types of threads
  - There are two types of Threads in java.
    - User threads are threads which are created by the application or user.
    - They are high priority threads.
    - JVM (Java Virtual Machine) will not exit until all user threads finish their execution.
    - JVM wait for these threads to finish their task. These threads are foreground threads.

Conti..

- Daemon threads are threads which are mostly created by the JVM.
- These threads always run-in background.
- These threads are used to perform some background tasks like garbage collection and house-keeping tasks.
- These threads are less priority threads.
- JVM will not wait for these threads to finish their execution.
- JVM will exit as soon as all user threads finish their execution.
- JVM doesn’t wait for daemon threads to finish their task.

Creating Threads

- Create a Thread by Implementing a Runnable Interface
  - If your class is intended to be executed as a thread, then you can achieve this by implementing a Runnable interface. You will need to follow three basic steps −
  - Step 1
    - As a first step, you need to implement a run() method provided by a Runnable interface. This method provides an entry point for the thread, and you will put your complete business logic inside this method. Following is a simple syntax of the run() method −
      - public void run( )

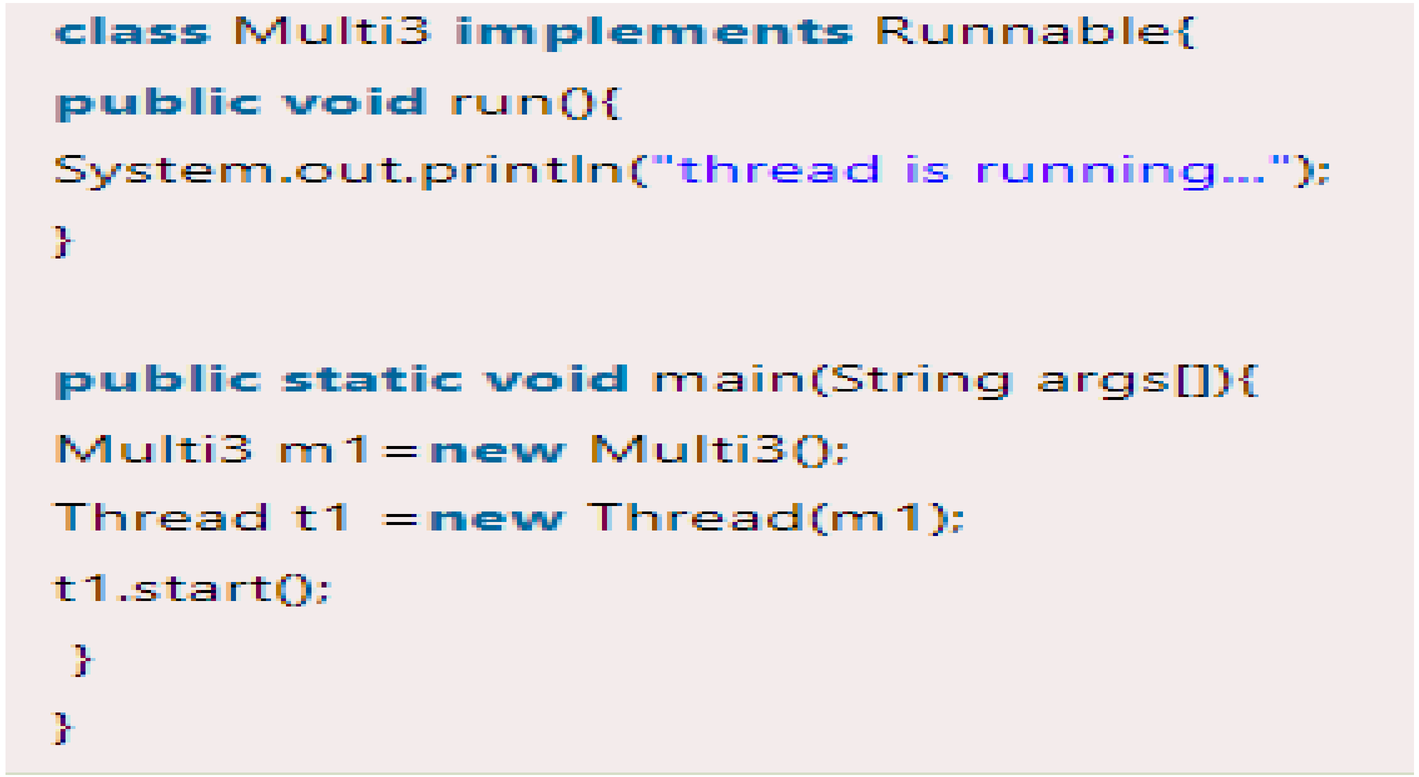
Conti…

- Step 2
  - As a second step, you will instantiate a Thread object using the following constructor −
    - Thread(Runnable threadObj, String threadName);
  - Where, threadObj is an instance of a class that implements the Runnable interface and threadName is the name given to the new thread.

- Step 3
  - Once a Thread object is created, you can start it by calling start() method, which executes a call to run( ) method. Following is a simple syntax of start() method −
    - void start();

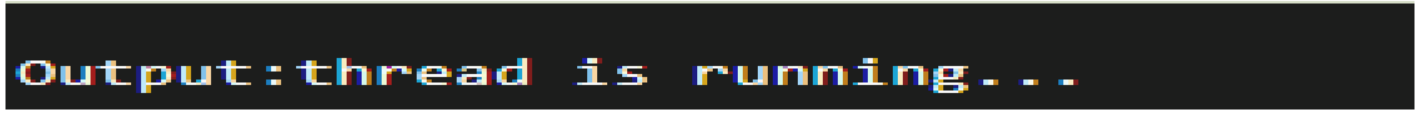
Creating Threads

Example:



```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}

public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);
t1.start();
 }
}
```



```
Output:thread is running...
```

Creating Threads

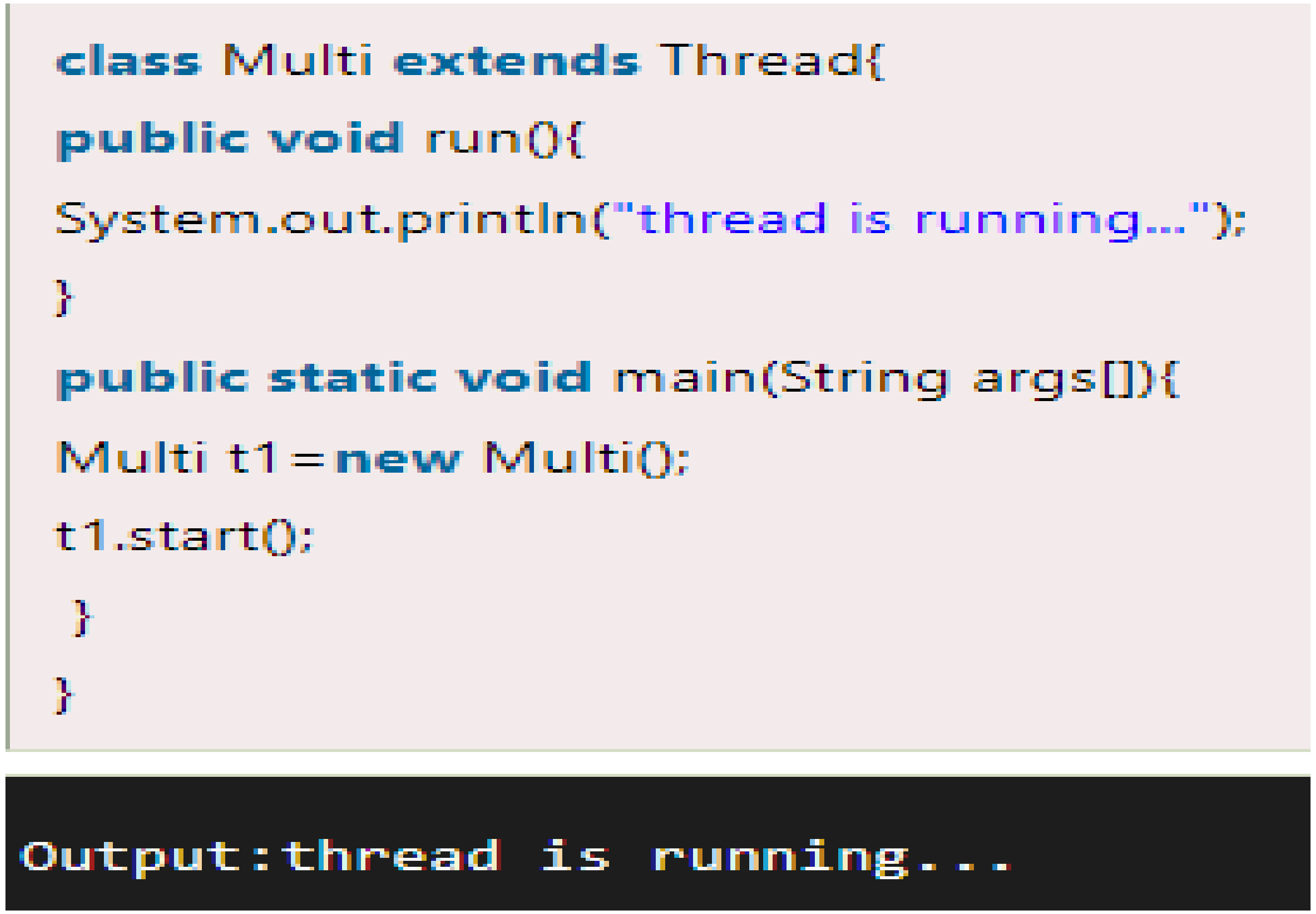
- Create a Thread by Extending a Thread Class
  - The second way to create a thread is to create a new class that extends Thread class using the following two simple steps. This approach provides more flexibility in handling multiple threads created using available methods in Thread class.
  - Step 1
    - You will need to override run( ) method available in Thread class. This method provides an entry point for the thread, and you will put your complete business logic inside this method. Following is a simple syntax of run() method –
      - public void run( )

Creating Threads

- Step 2
  - Once Thread object is created, you can start it by calling start() method, which executes a call to run( ) method. Following is a simple syntax of start() method-
    - void start( );

Creating Threads

Example:



```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

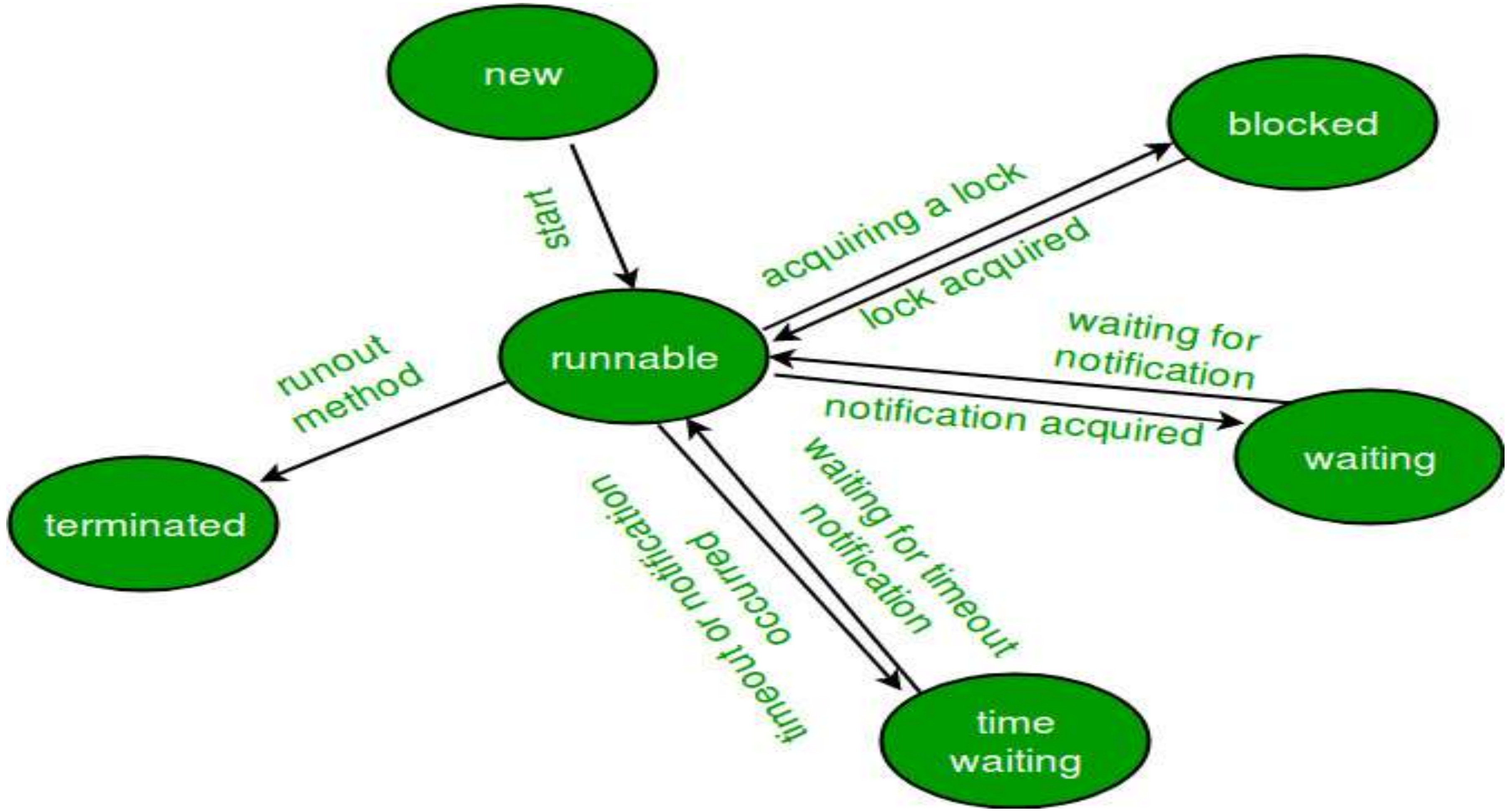
```
Output:thread is running...
```

Thread Lifecycle

- A thread in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:
  - New
  - Runnable
  - Blocked
  - Waiting
  - Timed Waiting
  - Terminated

Thread Lifecycle

- The diagram shown below represent various states of a thread at any instant of time.



Thread Lifecycle

- **New Thread:** When a new thread is created, it is in the new state.
  - The thread has not yet started to run when thread is in this state.
  - When a thread lies in the new state, it’s code is yet to be run and hasn’t started to execute.
- **Runnable State:** A thread that is ready to run is moved to runnable state.
  - In this state, a thread might be running or it might be ready run at any instant of time. It is the responsibility of the thread scheduler to give the thread, time to run.
  - A multi-threaded program allocates a fixed amount of time to each individual thread.
  - Each thread runs for a short while and then pauses and relinquishes the CPU to another thread, so that other threads can get a chance to run.
  - When this happens, all such threads that are ready to run, waiting for the CPU and the currently running thread lies in runnable state.

Cont….

- **Blocked/Waiting state:** When a thread is temporarily inactive, then it’s in one of the following states:
    - Blocked
    - Waiting
  - For example, when a thread is waiting for I/O to complete, it lies in the blocked state.
  - It’s the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread.
  - A thread in this state cannot continue its execution any further until it is moved to runnable state. Any thread in these states does not consume any CPU cycle.
  - A thread is in the blocked state when it tries to access a protected section of code that is currently locked by some other thread.

Thread Lifecycle

- When the protected section is unlocked, the schedule picks one of the thread which is blocked for that section and moves it to the runnable state.
- Whereas, a thread is in the waiting state when it waits for another thread on a condition.
- When this condition is fulfilled, the scheduler is notified and the waiting thread is moved to runnable state.
- If a currently running thread is moved to blocked/waiting state, another thread in the runnable state is scheduled by the thread scheduler to run.
- It is the responsibility of thread scheduler to determine which thread to run.

Thread Lifecycle

- **Timed Waiting:** A thread lies in timed waiting state when it calls a method with a time out parameter.
  - A thread lies in this state until the timeout is completed or until a notification is received.
  - For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.
- **Terminated State:** A thread terminates because of either of the following reasons:
  - Because it exists normally. This happens when the code of thread has entirely executed by the program.
  - Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.
- A thread that lies in a terminated state does no longer consumes any cycles of CPU.

Lecture 2

1. Threads: Introduction and Types
   - Thread Priorities
   - Daemon Thread
   - Runnable Class

Thread Priorities

- Every Java thread has a priority that helps the operating system determine the order in which threads are scheduled.
- Java thread priorities are in the range between MIN_PRIORITY (a constant of 1) and MAX_PRIORITY (a constant of 10). By default, every thread is given priority NORM_PRIORITY (a constant of 5).
- Threads with higher priority are more important to a program and should be allocated processor time before lower-priority threads.
- However, thread priorities cannot guarantee the order in which threads execute and are very much platform dependent.

Daemon Thread

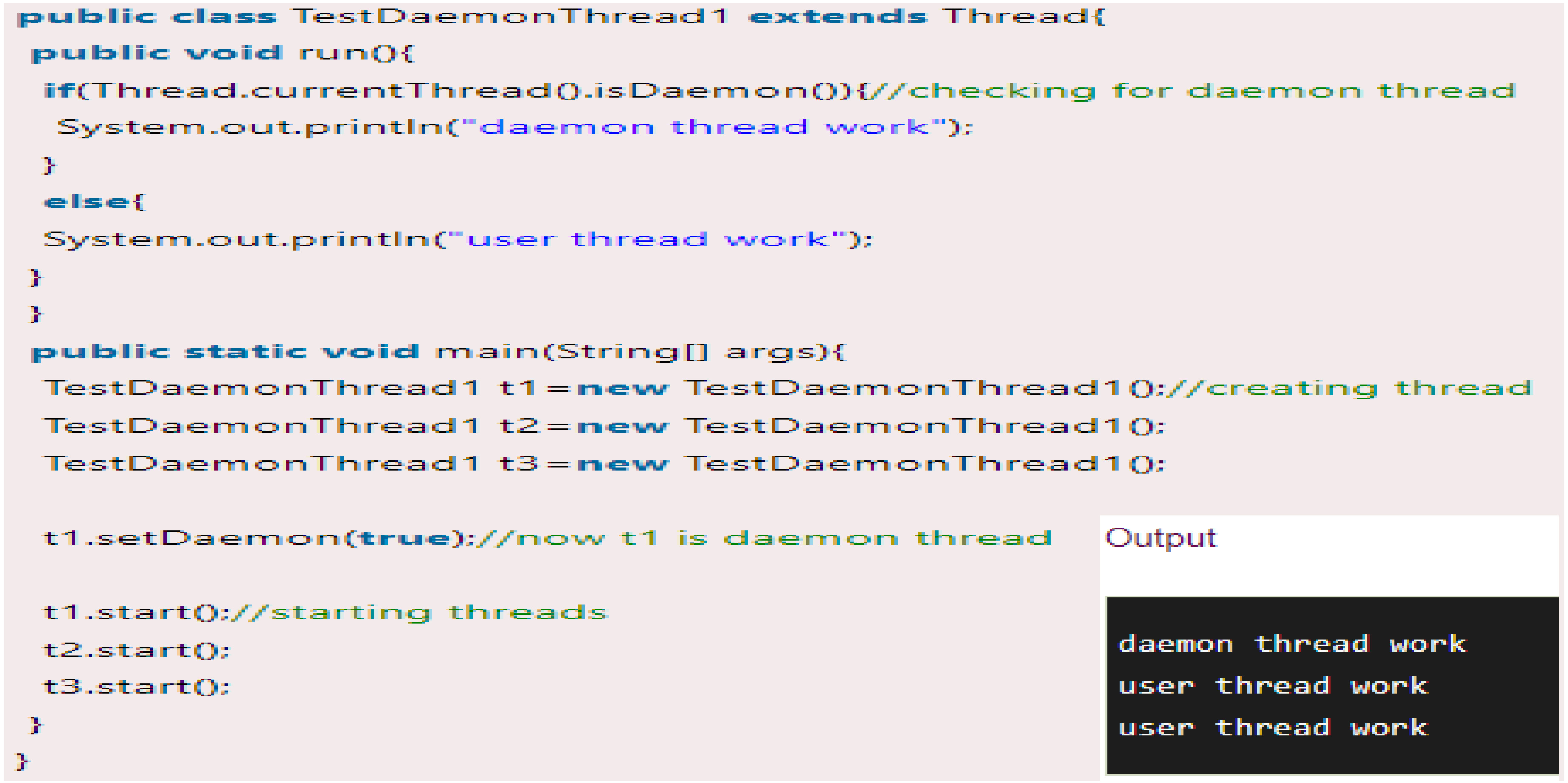
- **Daemon thread in java** is a service provider thread that provides services to the user thread.
- Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically.
- There are many java daemon threads running automatically e.g. gc, finalizer etc.
- Points to remember for Daemon Thread in Java
  - It provides services to user threads for background supporting tasks. It has no role in life than to serve user threads.
  - Its life depends on user threads.
  - It is a low priority thread.

Daemon Thread

- Methods for Java Daemon thread by Thread class
  - The java.lang.Thread class provides two methods for java daemon thread.

No.	Method	Description
1)	public void setDaemon(boolean status)	is used to mark the current thread as daemon thread or user thread.
2)	public boolean isDaemon()	is used to check that current is daemon.

Daemon Thread Example



```java
public class TestDaemonThread1 extends Thread{
 public void run(){
  if(Thread.currentThread().isDaemon()){//checking for daemon thread
   System.out.println("daemon thread work");
  }
  else{
  System.out.println("user thread work");
 }
 }
 public static void main(String[] args){
  TestDaemonThread1 t1=new TestDaemonThread1();//creating thread
  TestDaemonThread1 t2=new TestDaemonThread1();
  TestDaemonThread1 t3=new TestDaemonThread1();

  t1.setDaemon(true);//now t1 is daemon thread

  t1.start();//starting threads
  t2.start();
  t3.start();
 }
}
```

Output

```
daemon thread work
user thread work
user thread work
```

Runnable Class

- java.lang.Runnable is an interface that is to be implemented by a class whose instances are intended to be executed by a thread.
- There are two ways to start a new Thread – Subclass Thread and implement Runnable.
- There is no need of subclassing Thread when a task can be done by overriding only run() method of Runnable.
- Create a Runnable implementer and implement run() method.
- Instantiate Thread class and pass the implementer to the Thread, Thread has a constructor which accepts Runnable instance.
- Invoke start() of Thread instance, start internally calls run() of the implementer. Invoking start(), creates a new Thread which executes the code written in run().
- Calling run() directly doesn't create and start a new Thread, it will run in the same thread. To start a new line of execution, call start() on the thread.

Runnable Class Example

```
public class RunnableDemo {

    public static void main(String[] args)
    {
        System.out.println("Main thread is- "
                        + Thread.currentThread().getName());
        Thread t1 = new Thread(new RunnableDemo().new RunnableImpl());
        t1.start();
    }

    private class RunnableImpl implements Runnable {

        public void run()
        {
            System.out.println(Thread.currentThread().getName()
                             + ", executing run() method!");
        }
    }
}
```

Output:

```
Main thread is- main
Thread-0, executing run() method!
```

Lecture 3

1. Threads: Introduction and Types
   - Synchronizing Threads

Synchronized Threads

- The process of allowing multiple threads to modify an object in a sequence is called synchronization.
- We can allow multiple threads to modify the objects sequentially only by executing that objects' mutator methods logic in sequence from multiple threads.
- This is possible by using an object locking concept.
- Thread Synchronization is a process of allowing only one thread to use the object when multiple threads are trying to use the object at the same time.
- To achieve this Thread Synchronization, we must use a java keyword or modifier called "synchronized".
- Synchronization in java is the capability to control the access of multiple threads to any shared resource. Java Synchronization is a better option where we want to allow only one thread to access the shared resource.

Synchronized Threads

- General Syntax:

```
synchronized(objectidentifier)
{
        // Access shared variables and other shared resources;
}
```

- Why use Synchronization?
    - The synchronization is mainly used to :
        - If you start with at least two threads inside a program, there might be a chance when multiple threads attempt to get to the same resource.
        - It can even create an unexpected outcome because of concurrency issues.

Synchronized Threads

- Types of Synchronization
  - There are basically two types of synchronization available. They are:
    - Process Synchronization: It means sharing system resources by processes in such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data.
    - Thread Synchronization: It means that every access to data shared between threads is protected so that when any thread starts operation on the shared data, no other thread is allowed access until the first thread is done.

Synchronized Threads

- Locks in Java
  - Synchronization is built around an internal entity known as the lock or monitor lock. (The API specification often refers to this entity simply as a “monitor.”)
  - Locks play a role in both aspects of synchronization: enforcing exclusive access to an object’s state and establishing happens-before relationships that are essential to visibility.
  - Every object has a lock associated with it.
  - By convention, a thread that needs exclusive and consistent access to an object’s fields has to acquire the object’s lock before accessing them, and then release the lock when it’s done with them.
  - A thread is said to own the lock between the time it has acquired the lock and released the lock.
  - As long as a thread owns a lock, no other thread can acquire the same lock. The other thread will block when it attempts to acquire the lock.

Lecture 4

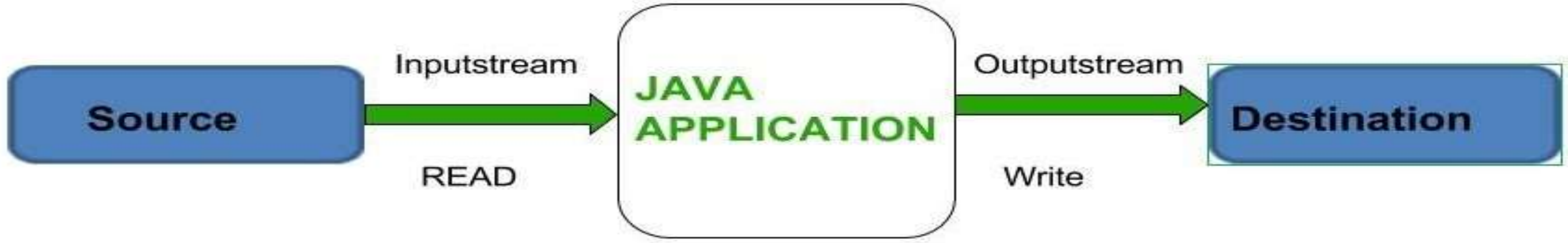
1. I/O Stream: Introduction and Types
   - Common I/O Stream Operations
   - Interaction with I/O Streams Classes

Topic Objective

- To understand the difference between binary files and text files
- To learn how to read and save data in a file

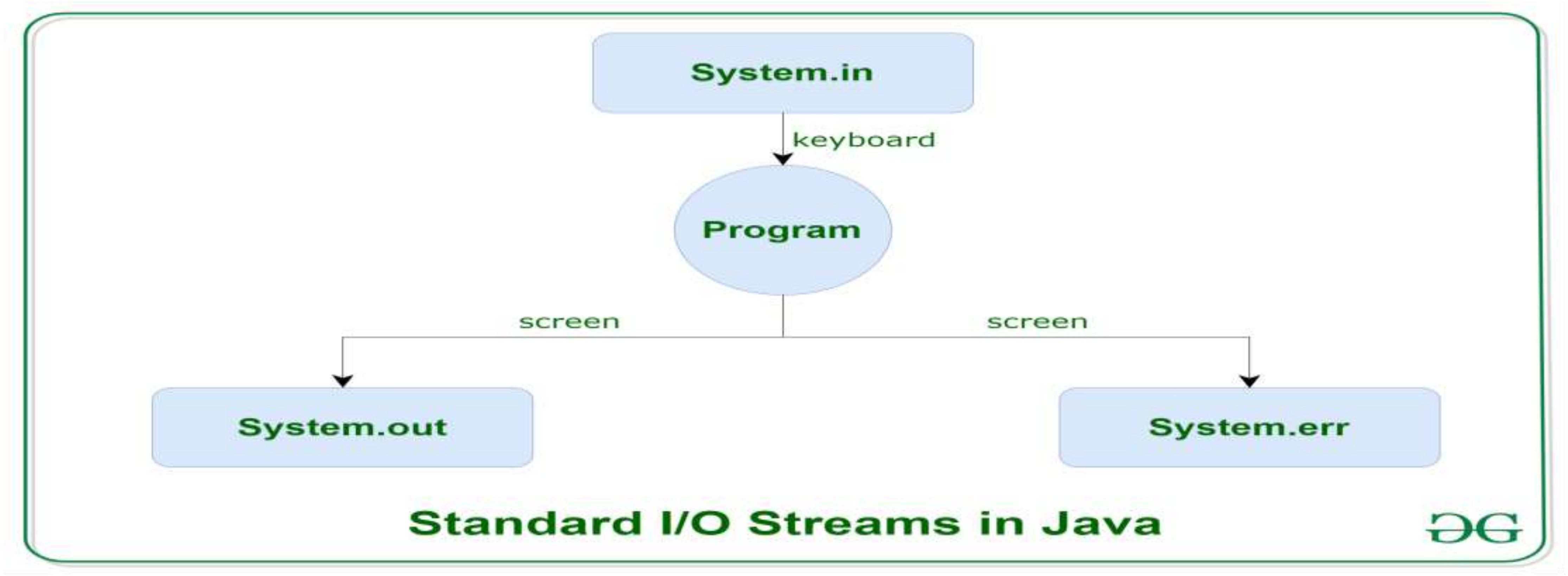
I/O Stream: Introduction

- Java brings various Streams with its I/O package that helps the user to perform all the input-output operations.
- These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations.



I/O Stream: Introduction

- Before exploring various input and output streams lets look at 3 standard or default streams that Java must provide which are also most common in use:



I/O Stream: Introduction

- System.in: This is the standard input stream that is used to read characters from the keyboard or any other standard input device.
- System.out: This is the standard output stream that is used to produce the result of a program on an output device like the computer screen.
- Here is a list of the various print functions that we use to output statements:
  - print(): This method in Java is used to display a text on the console.
  - This text is passed as the parameter to this method in the form of String.
  - This method prints the text on the console and the cursor remains at the end of the text at the console.
  - The next printing takes place from just here.
    - Syntax:
    - System.out.print(parameter);

I/O Stream: Introduction

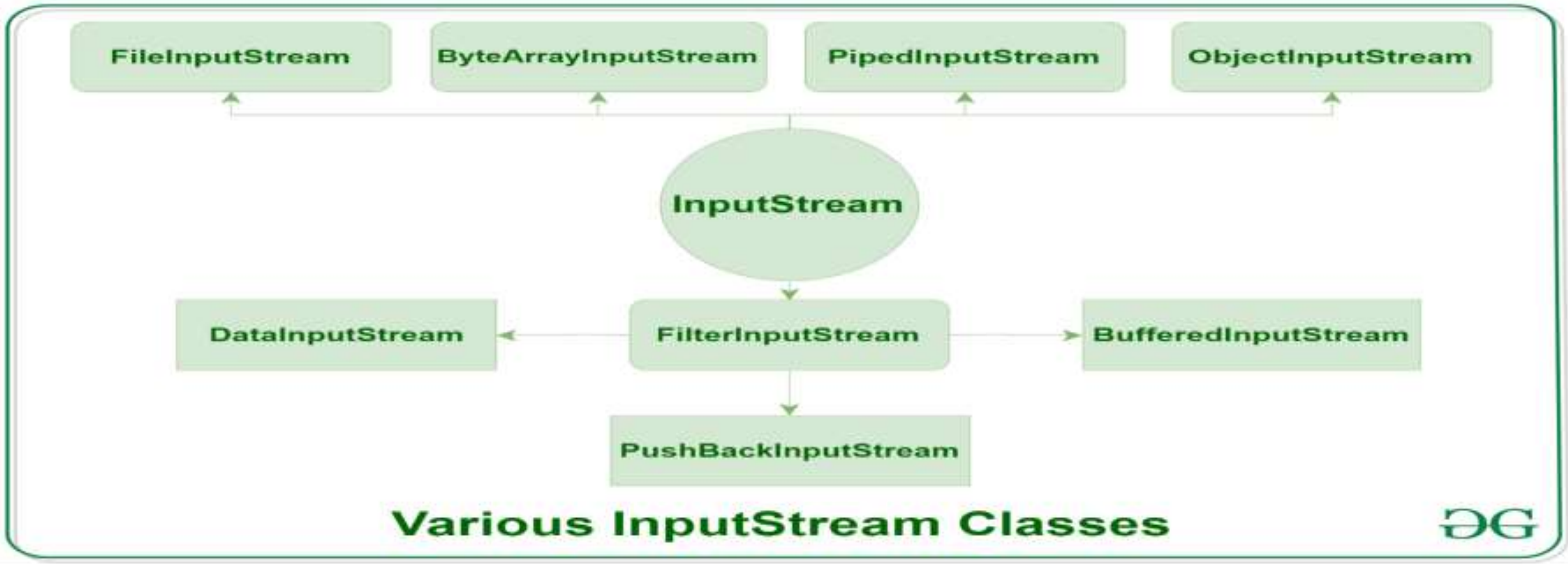
- println(): This method in Java is also used to display a text on the console.
- It prints the text on the console and the cursor moves to the start of the next line at the console.
- The next printing takes place from the next line.
  - Syntax:
  - System.out.println(parameter);
- printf()**:** This is the easiest of all methods as this is similar to printf in C.
- Note that System.out.print() and System.out.println() take a single argument, but printf() may take multiple arguments.
- This is used to format the output in Java.

I/O Stream: Introduction

- System.err: This is the standard error stream that is used to output all the error data that a program might throw, on a computer screen or any standard output device.
- This stream also uses all the 3 above-mentioned functions to output the error data:
- print()
- println()
- printf()

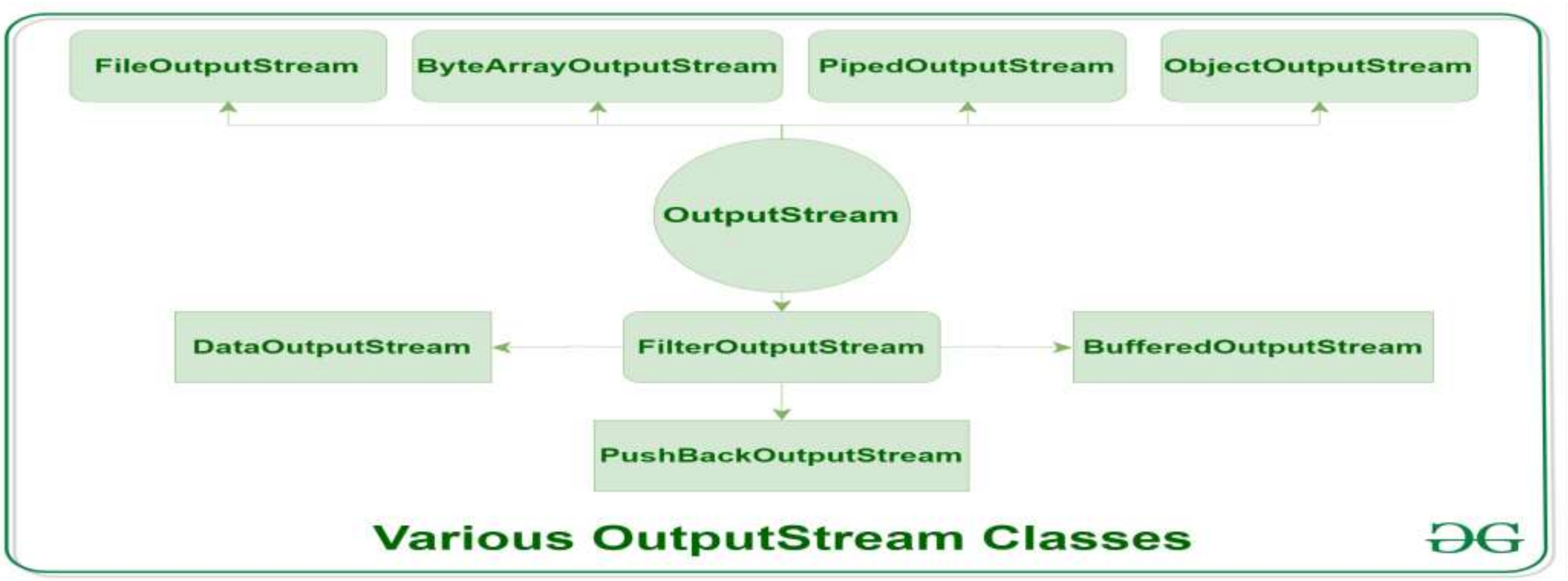
Types of Streams

- Depending on the type of operations, streams can be divided into two primary classes:
    - Input Stream: These streams are used to read data that must be taken as an input from a source array or file or any peripheral device. For eg., FileInputStream, BufferedInputStream, ByteArrayInputStream etc.



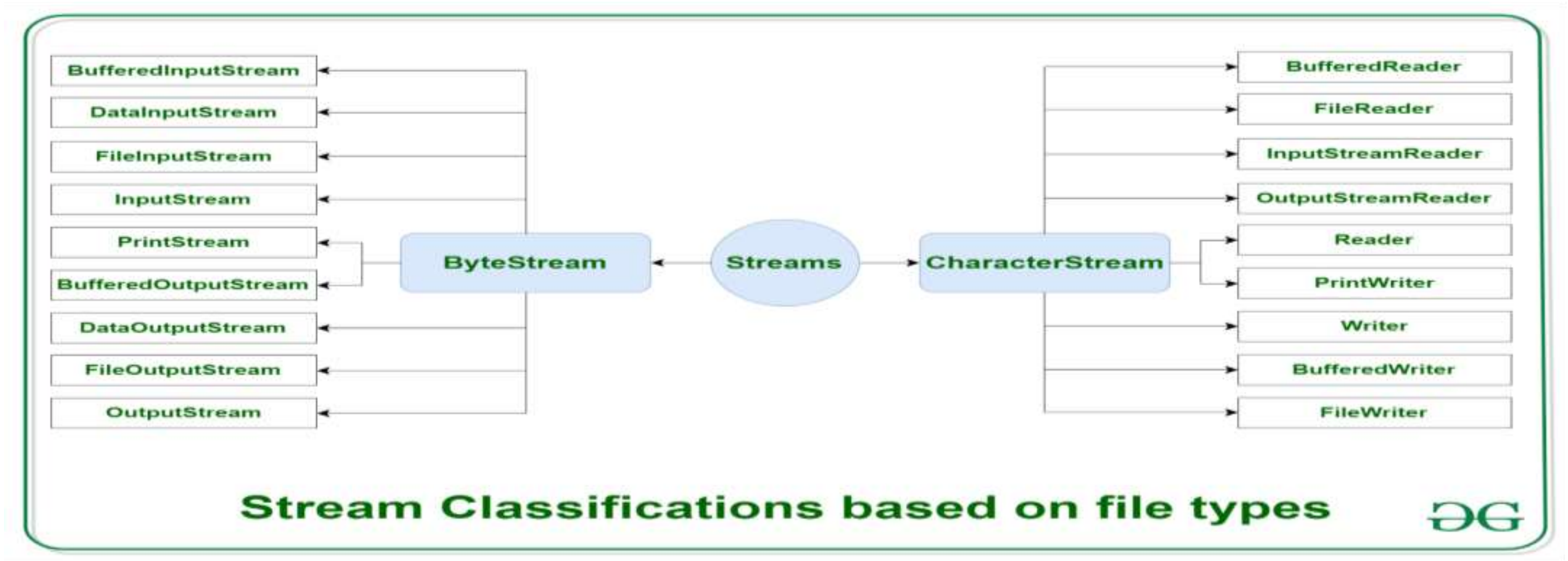
Types of Streams

- Output Stream: These streams are used to write data as outputs into an array or file or any output peripheral device. For eg., FileOutputStream, BufferedOutputStream, ByteArrayOutputStream etc.



Types of Streams

- Depending on the types of file, Streams can be divided into two primary classes which can be further divided into other classes as can be seen through the diagram below followed by the explanations.



Types of Streams

- ByteStream: This is used to process data byte by byte (8 bits).
- Though it has many classes, the FileInputStream and the FileOutputStream are the most popular ones.
- The FileInputStream is used to read from the source and FileOutputStream is used to write to the destination.
- CharacterStream: In Java, characters are stored using Unicode conventions.
- Character stream automatically allows us to read/write data character by character.
- Though it has many classes, the FileReader and the FileWriter are the most popular ones.
- FileReader and FileWriter are character streams used to read from the source and write to the destination respectively.

Common I/O Stream Operations

- Useful methods of Output Stream

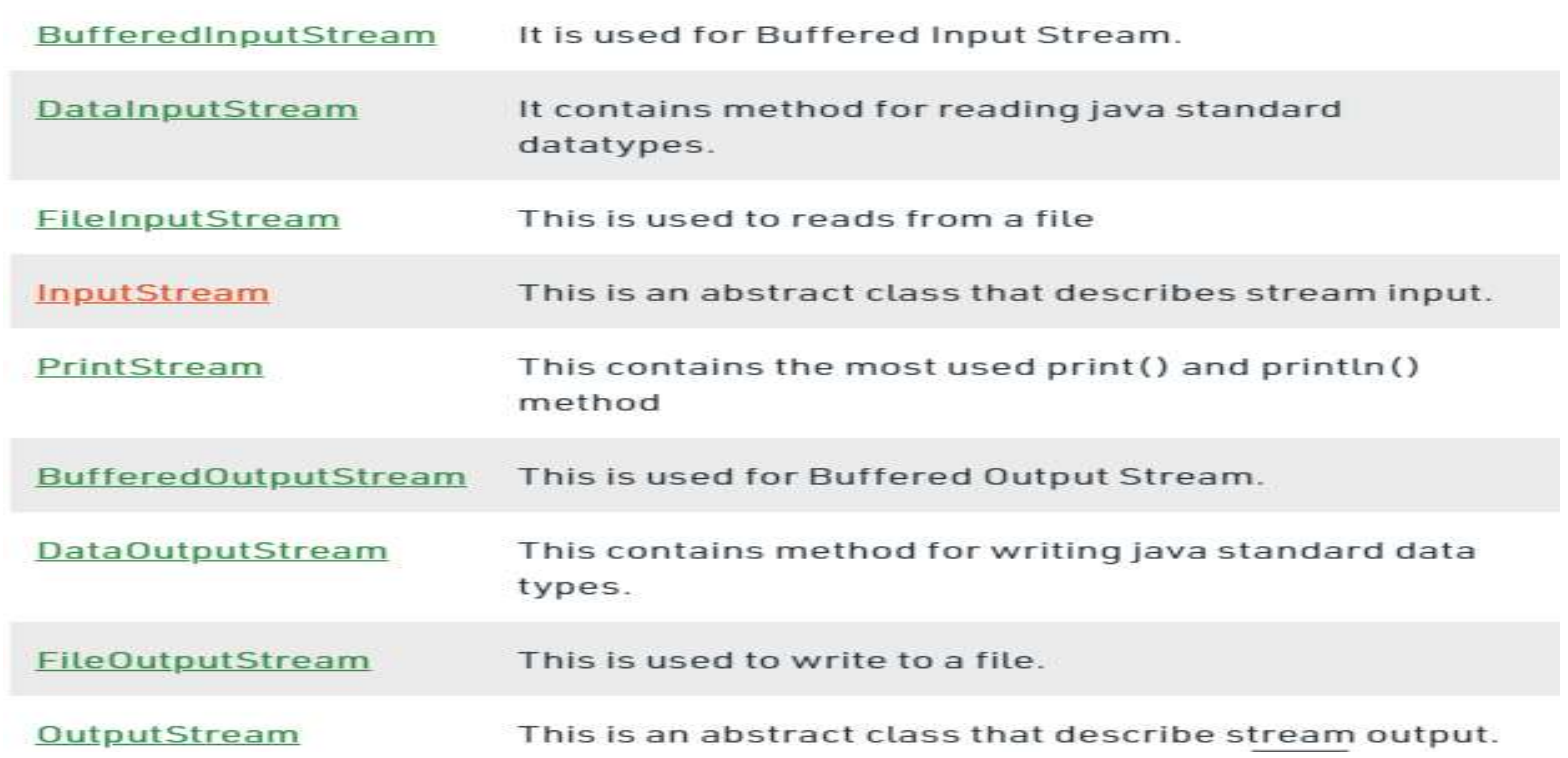
	Description
1) public void write(int)throws IOException	is used to write a byte to the current output stream.
2) public void write(byte[])throws IOException	is used to write an array of byte to the current output stream.
3) public void flush()throws IOException	flushes the current output stream.
4) public void close()throws IOException	is used to close the current output stream.

Conti…..

- **Useful methods of Input Stream**

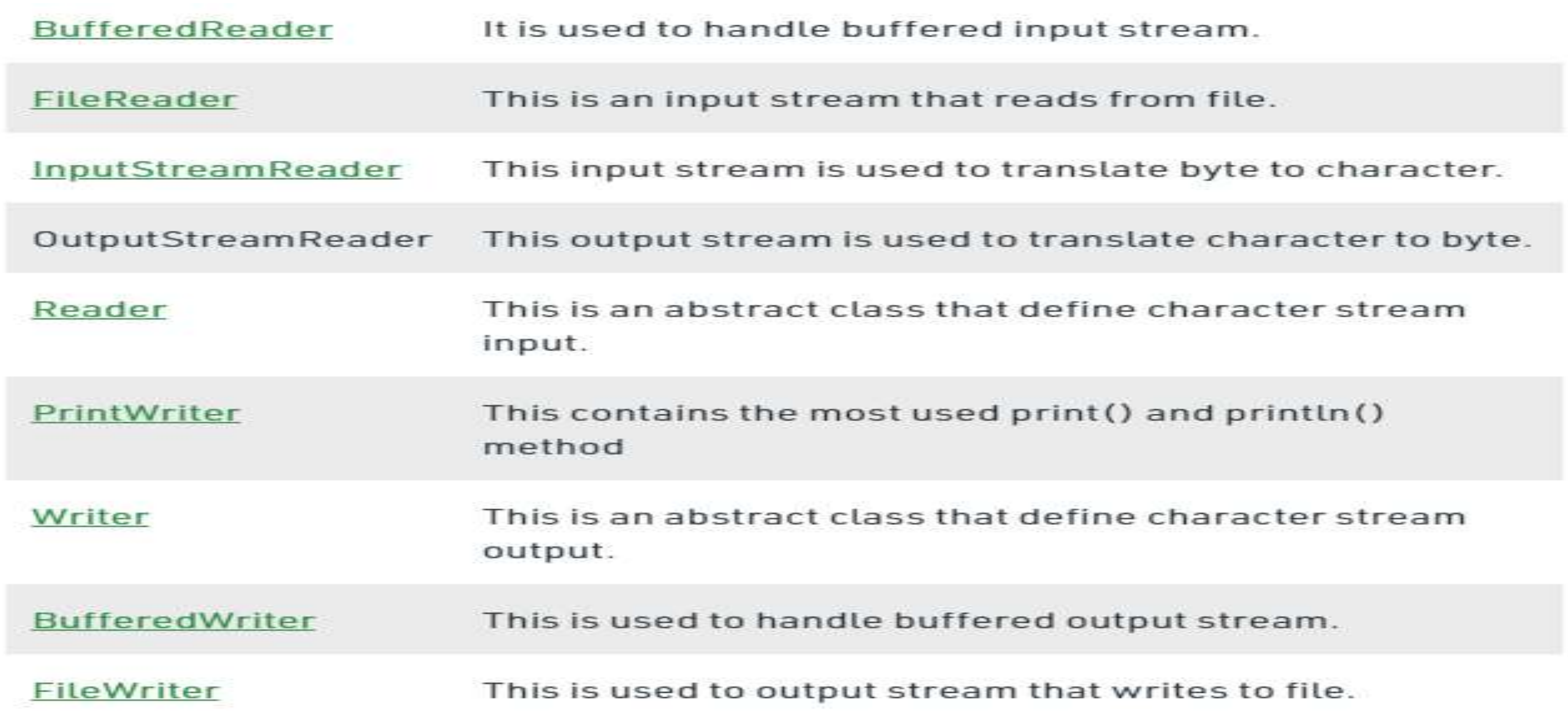
Method	Description
1) public abstract int read()throws IOException	reads the next byte of data from the input stream. It returns -1 at the end of the file.
2) public int available()throws IOException	returns an estimate of the number of bytes that can be read from the current input stream.
3) public void close()throws IOException	is used to close the current input stream.

Interaction with I/O Streams Classes



BufferedInputStream	It is used for Buffered Input Stream.
DataInputStream	It contains method for reading java standard datatypes.
FileInputStream	This is used to reads from a file
InputStream	This is an abstract class that describes stream input.
PrintStream	This contains the most used print() and println() method
BufferedOutputStream	This is used for Buffered Output Stream.
DataOutputStream	This contains method for writing java standard data types.
FileOutputStream	This is used to write to a file.
OutputStream	This is an abstract class that describe stream output.

Interaction with I/O Streams Classes



BufferedReader	It is used to handle buffered input stream.
FileReader	This is an input stream that reads from file.
InputStreamReader	This input stream is used to translate byte to character.
OutputStreamReader	This output stream is used to translate character to byte.
Reader	This is an abstract class that define character stream input.
PrintWriter	This contains the most used print() and println() method
Writer	This is an abstract class that define character stream output.
BufferedWriter	This is used to handle buffered output stream.
FileWriter	This is used to output stream that writes to file.

Lecture 5

Annotations: Introduction

- Custom Annotations
- Applying Annotations

Annotations: Introduction

- Annotations are used to provide supplement information about a program.
- Annotations start with ‘@’.
- Annotations do not change action of a compiled program.
- Annotations help to associate metadata (information) to the program elements i.e. instance variables, constructors, methods, classes, etc.
- Annotations are not pure comments as they can change the way a program is treated by compiler.

Annotations: Introduction (CO4)

- Built-In Java Annotations used in Java code
- @Override
- @SuppressWarnings
- @Deprecated

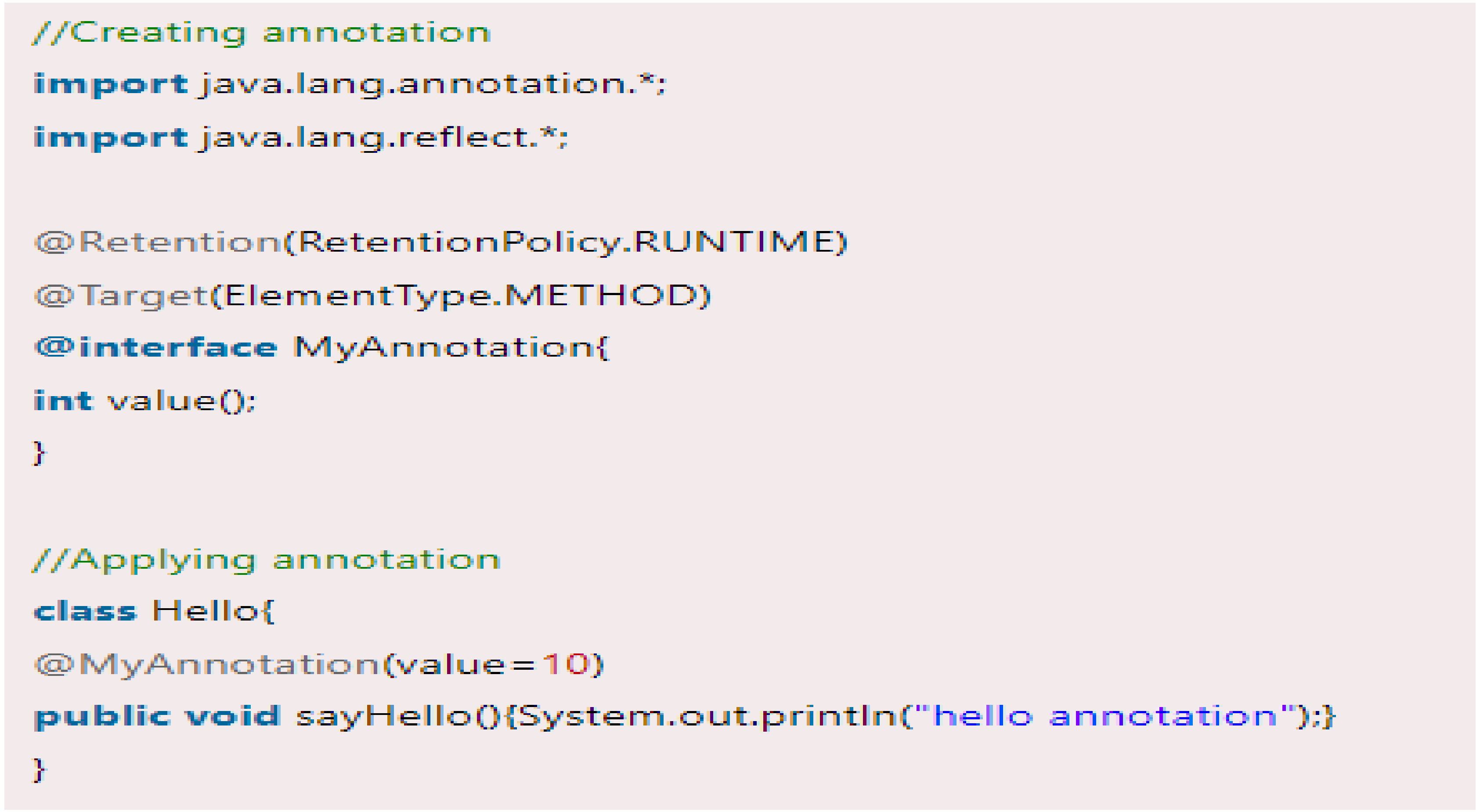
- @Override
  - @Override annotation assures that the subclass method is overriding the parent class method. If it is not so, compile time error occurs.
- @SuppressWarnings
  - @SuppressWarnings annotation is used to suppress warnings issued by the compiler.
- @Deprecated
  - @Deprecated annotation marks that this method is deprecated so compiler prints warning. It informs user that it may be removed in the future versions. So, it is better not to use such methods.

Custom Annotations

- Java Custom annotations or Java User-defined annotations are easy to create and use. The @interface element is used to declare an annotation. For example:
  - @interface MyAnnotation{}
- Here, MyAnnotation is the custom annotation name.
- Points to remember for java custom annotation signature
  - Method should not have any throws clauses
  - Method should return one of the following: primitive data types, String, Class, enum or array of these data types.
  - Method should not have any parameter.
  - We should attach @ just before interface keyword to define annotation.
  - It may assign a default value to the method.

Applying Annotations

- Example of custom annotation: creating, applying and accessing annotation

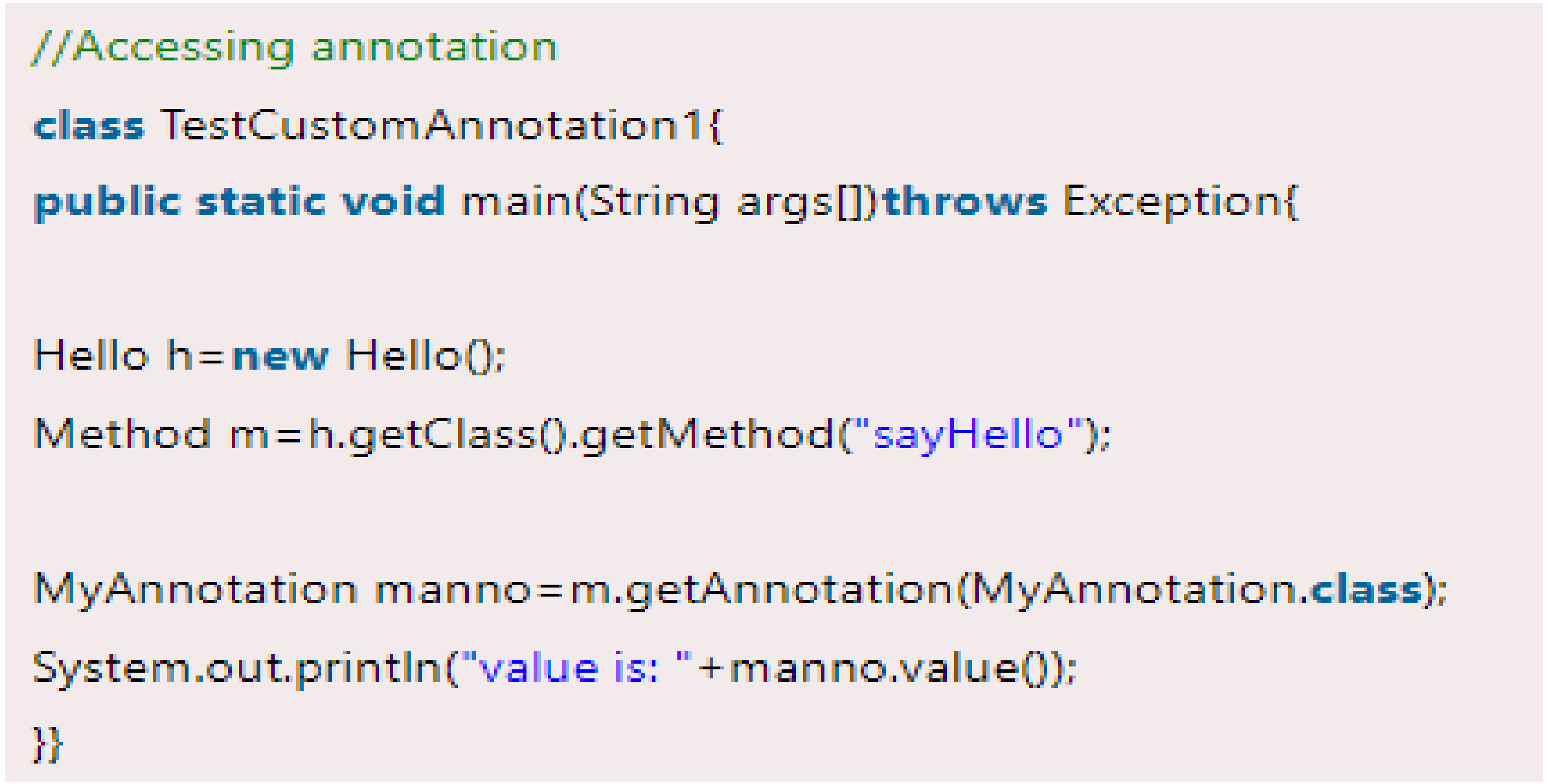


```
//Creating annotation
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface MyAnnotation{
int value();
}

//Applying annotation
class Hello{
@MyAnnotation(value=10)
public void sayHello(){System.out.println("hello annotation");}
}
```

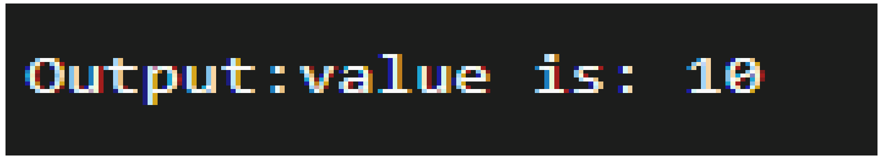
Applying Annotations



```
//Accessing annotation
class TestCustomAnnotation1{
public static void main(String args[])throws Exception{

Hello h=new Hello();
Method m=h.getClass().getMethod("sayHello");

MyAnnotation manno=m.getAnnotation(MyAnnotation.class);
System.out.println("value is: "+manno.value());
}}
```



```
Output:value is: 10
```

DAILY QUIZ

1. What is Multithreading? What are the ways to create multiple threads in java. [CO4]
2. Explain about Thread Life Cycle. [CO4]
3. Define Daemon Threads? Explain with an example. [CO4]
4. Write a java program to implement join() method in multithreading. [CO4]
5. What are the states of a thread? [CO4]
6. What are the threads will start, when you start the java program? [CO4]
7. What are the different identifier states of a Thread? [CO4]
8. Why do threads block on I/O? [CO4]
9. What is synchronization and why is it important? [CO4]

NIET Greater Noida GET FUTURE READY Estd.2001 AN AUTONOMOUS INSTITUTE

Weekly Assignment

Q1. Explain Threads and types of threads in java.[CO4]

Q2. Explain Thread Life Cycle in detail.[CO4]

Q3. Explain Thread Creation in java with suitable example.[CO4]

Q4. What is Daemon Thread? Give its example.[CO4]

Q5. Explain input output stream operations in java.[CO4]

Q6. What are Custom Annotations? Explain how to create custom annotations in java with suitable example.[CO4]

Q7. Explain various types of Output Streams in java.[CO4]

Q8. What do you understand by Annotations? Explain various built in java annotations in java.[CO4]

Q9.Explain various methods of input stream in detail.[CO4]

Q10. Explain various output streams methods in detail. [CO4]

TOPIC LINKS

- https://www.youtube.com/watch?v=qQVqfvs3p48
- https://www.youtube.com/watch?v=TCd8QIS-2KI
- https://www.youtube.com/watch?v=vd9nJK22BwU

MCQ

Q1. Which of these method of Thread class is used to find out the priority given to a thread?[CO4]
a) get()
b) ThreadPriority()
c) getPriority()
d) getThreadPriority()

Q2. Which of these method of Thread class is used to Suspend a thread for a period of time?[CO4]
a) sleep()
b) terminate()
c) suspend()
d) stop()

MCQ

Q3.Which function of pre defined class Thread is used to check weather current thread being checked is still running?[CO4]

a) isAlive()

b) Join()

c) isRunning()

d) Alive()

Q4. In java a thread can be created by ....[CO4]

A. Extending the thread class.

B.Implementing Runnable interface.

C.Both of the above

D.None of these

MCQ

Q5.Thread synchronization in a process will be required when[CO4]

A.All threads sharing the same address space

B.All threads sharing the same global variables

C.All threads sharing the same files

D.All

Q6.Which thread will be executed first if two threads have same priority[CO4]

A. They will fall in starvation and none will be executed.

B. Both will be executed simultaneously

C. It depends upon operating system

D.They will be executed on first come first serve basis

MCQ

Q7.The life cycle of a thread in java is controlled by[CO4]

A.JRE

B.JDK

C.JVM

D.None

Q8.Which method is used to get current running thread object?[CO4]

A.runningThread()

B.currentThread()

C.runnableThread()

D.None

MCQs

Q9.Which version of Java introduced annotation?[CO4]

a) Java 5

b) Java 6

c) Java 7

d) Java 8

Q10.Annotations which are applied to other annotations are called meta annotations.[CO4]

a) True

b) False

Q11.Which of these is used to perform all input & output operations in Java?[CO4]

a) streams

b) Variables

c) classes

d) Methods

MCQ

Q12.Which of these is a type of stream in Java?[CO4]

a) Integer stream

b) Short stream

c) Byte stream

d) Long stream

Q13.Which of these classes are used by Byte streams for input and output operation?[CO4]

a) InputStream

b) InputOutputStream

c) Reader

d) All of the mentioned

Q14. Which of these classes are used by character streams for input and output operations?[CO4]

a) InputStream

b) Writer

c) ReadStream

d) InputOutputStream

MCQ

Q15.Daemon thread runs in [CO4]

A. Background

B. Foreground

C. Both

D.None

Q16. Which method is used to create a daemon thread?[CO4]

A. setDaemon(boolean value)

B. setThread(boolean value)

C. createDaemon(boolean value)

D. makeDeamon(boolean value);

MCQ

Q17. Which will contain the body of the thread in Java?[CO4]

A. Start()

B. Run()

C. Main()

D. Execute()

Q18. To create a daemon thread [CO4]

A. First thread setDaemon() is called then start()

B. First thread start() is called then setDaemon()

C. Call thread start() and setDaemon() in any order

D.All correct

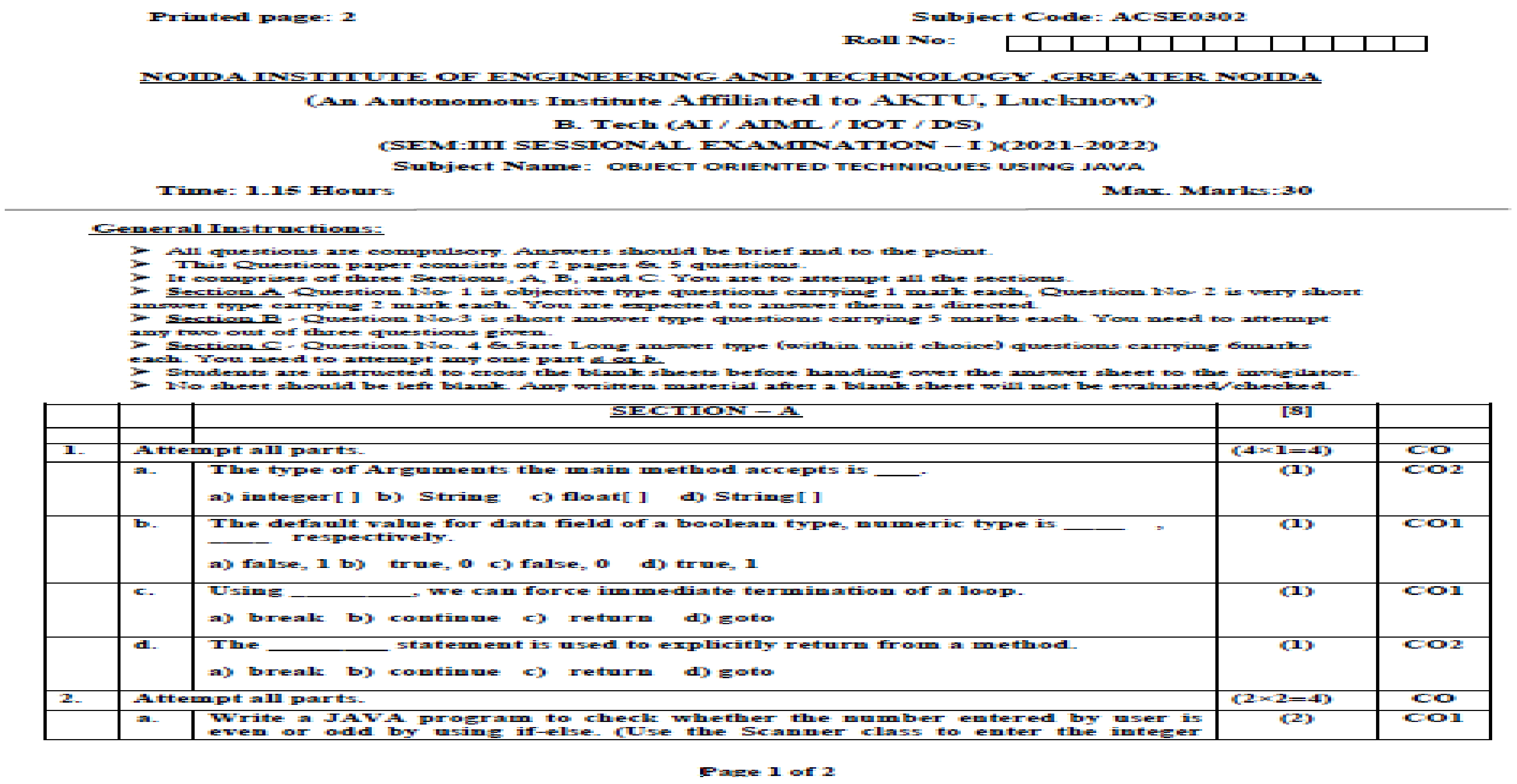
Glossary Questions

1. **Attempt all the parts. Pick the correct option from glossary.** [CO4]

   i) JVM ii) Run() iii) Background iv) streams

   a) __________method contain the body of the thread in java.
   b) By using __________ input and output operations are performed in java.
   c) Daemon threads run in the ___________.
   d) Thread Lifecycle in java is controlled by ___________.

Sessional Paper-1



Printed page: 2 Subject Code: ACSE0302

Roll No: | | | | | | | | | | | | | |

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY ,GREATER NOIDA

(An Autonomous Institute Affiliated to AKTU, Lucknow)

B. Tech (AI / AIML / IOT / DS)

(SEM:III SESSIONAL EXAMINATION – I)(2021-2022)

Subject Name: OBJECT ORIENTED TECHNIQUES USING JAVA

Time: 1.15 Hours Max. Marks:30

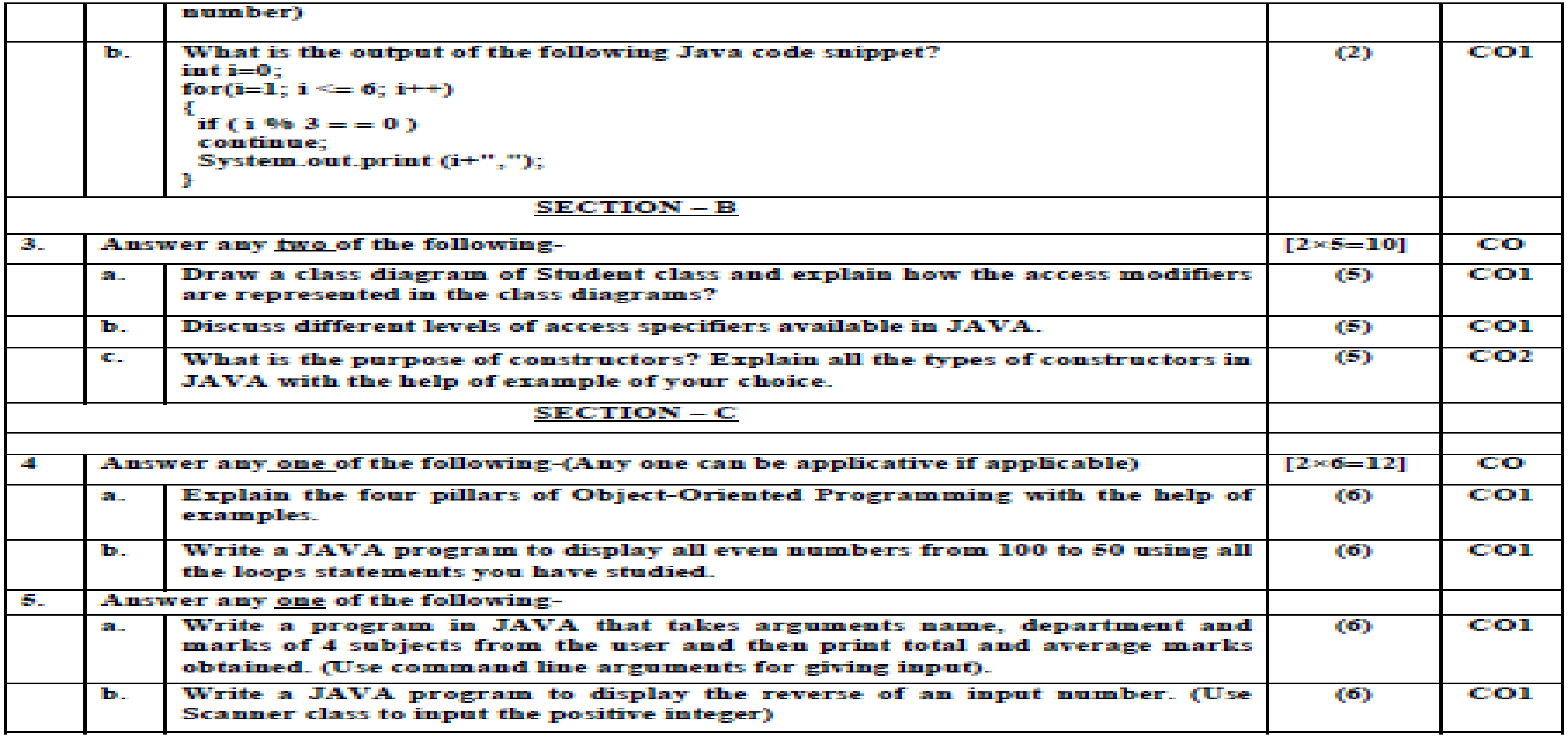
General Instructions:

- All questions are compulsory. Answers should be brief and to the point.
- This Question paper consists of 2 pages & 5 questions.
- It comprises of three Sections, A, B, and C. You are to attempt all the sections.
- Section A -Question No- 1 is objective type questions carrying 1 mark each, Question No- 2 is very short answer type carrying 2 mark each. You are expected to answer them as directed.
- Section B - Question No-3 is short answer type questions carrying 5 marks each. You need to attempt any two out of three questions given.
- Section C - Question No. 4 & 5are Long answer type (within unit choice) questions carrying 6marks each. You need to attempt any one part a or b.
- Students are instructed to cross the blank sheets before handing over the answer sheet to the invigilator.
- No sheet should be left blank. Any written material after a blank sheet will not be evaluated/checked.

		SECTION – A	[8]	
1.	Attempt all parts.		(4×1=4)	CO
	a.	The type of Arguments the main method accepts is ____. a) integer[ ] b) String c) float[ ] d) String[ ]	(1)	CO2
	b.	The default value for data field of a boolean type, numeric type is ______ , ______ respectively. a) false, 1 b) true, 0 c) false, 0 d) true, 1	(1)	CO1
	c.	Using ____________, we can force immediate termination of a loop. a) break b) continue c) return d) goto	(1)	CO1
	d.	The ____________ statement is used to explicitly return from a method. a) break b) continue c) return d) goto	(1)	CO2
2.	Attempt all parts.		(2×2=4)	CO
	a.	Write a JAVA program to check whether the number entered by user is even or odd by using if-else. (Use the Scanner class to enter the integer	(2)	CO1

Page 1 of 2

Conti….

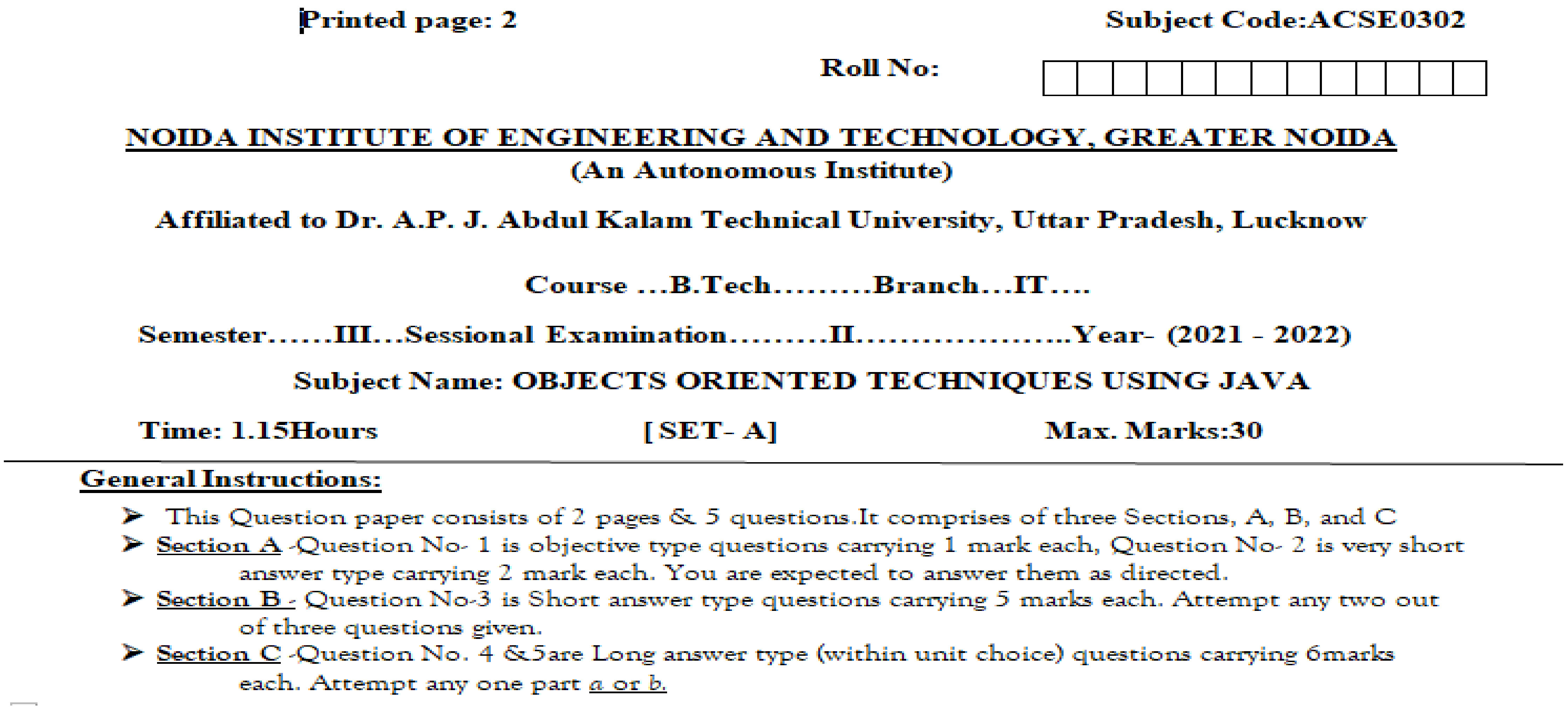


		number)		
	b.	What is the output of the following Java code snippet?	(2)	CO1

```
int i=0;
for(i=1; i <= 6; i++)
{
 if ( i % 3 = = 0 )
 continue;
 System.out.print (i+",");
}
```

		SECTION – B		
3.	Answer any two of the following-		[2×5=10]	CO
	a.	Draw a class diagram of Student class and explain how the access modifiers are represented in the class diagrams?	(5)	CO1
	b.	Discuss different levels of access specifiers available in JAVA.	(5)	CO1
	c.	What is the purpose of constructors? Explain all the types of constructors in JAVA with the help of example of your choice.	(5)	CO2
		SECTION – C		
4	Answer any one of the following-(Any one can be applicative if applicable)		[2×6=12]	CO
	a.	Explain the four pillars of Object-Oriented Programming with the help of examples.	(6)	CO1
	b.	Write a JAVA program to display all even numbers from 100 to 50 using all the loops statements you have studied.	(6)	CO1
5.	Answer any one of the following-			
	a.	Write a program in JAVA that takes arguments name, department and marks of 4 subjects from the user and then print total and average marks obtained. (Use command line arguments for giving input).	(6)	CO1
	b.	Write a JAVA program to display the reverse of an input number. (Use Scanner class to input the positive integer)	(6)	CO1

Sessional Paper-2



Printed page: 2　　　　　　　　　　　　Subject Code:ACSE0302

Roll No: | | | | | | | | | | | | |

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA

(An Autonomous Institute)

Affiliated to Dr. A.P. J. Abdul Kalam Technical University, Uttar Pradesh, Lucknow

Course ...B.Tech.........Branch...IT....

Semester......III...Sessional Examination.........II...................Year- (2021 - 2022)

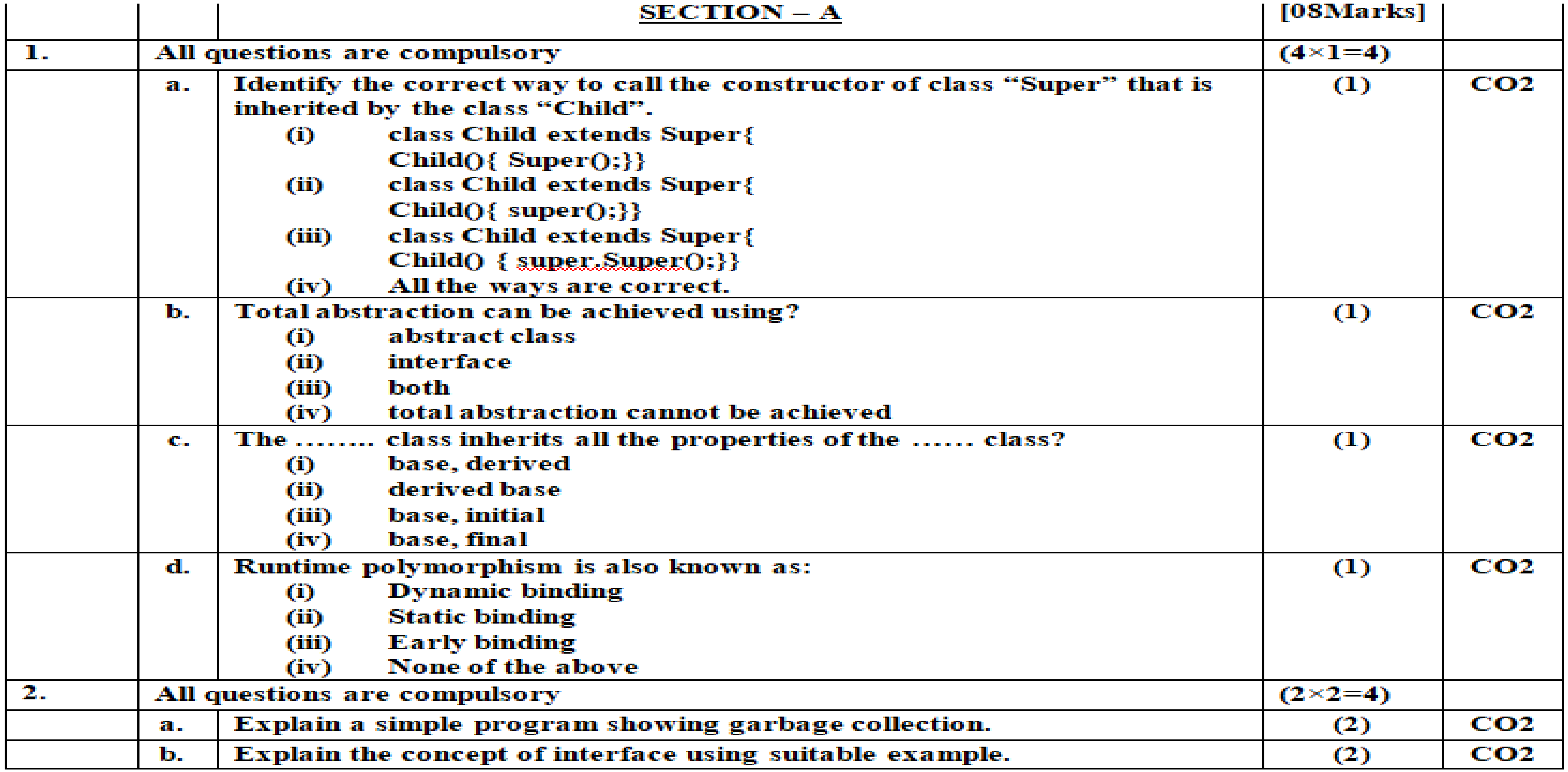
Subject Name: OBJECTS ORIENTED TECHNIQUES USING JAVA

Time: 1.15Hours　　　　[SET- A]　　　　Max. Marks:30

General Instructions:

- This Question paper consists of 2 pages & 5 questions.It comprises of three Sections, A, B, and C
- Section A -Question No- 1 is objective type questions carrying 1 mark each, Question No- 2 is very short answer type carrying 2 mark each. You are expected to answer them as directed.
- Section B - Question No-3 is Short answer type questions carrying 5 marks each. Attempt any two out of three questions given.
- Section C -Question No. 4 &5are Long answer type (within unit choice) questions carrying 6marks each. Attempt any one part a or b.

Conti….

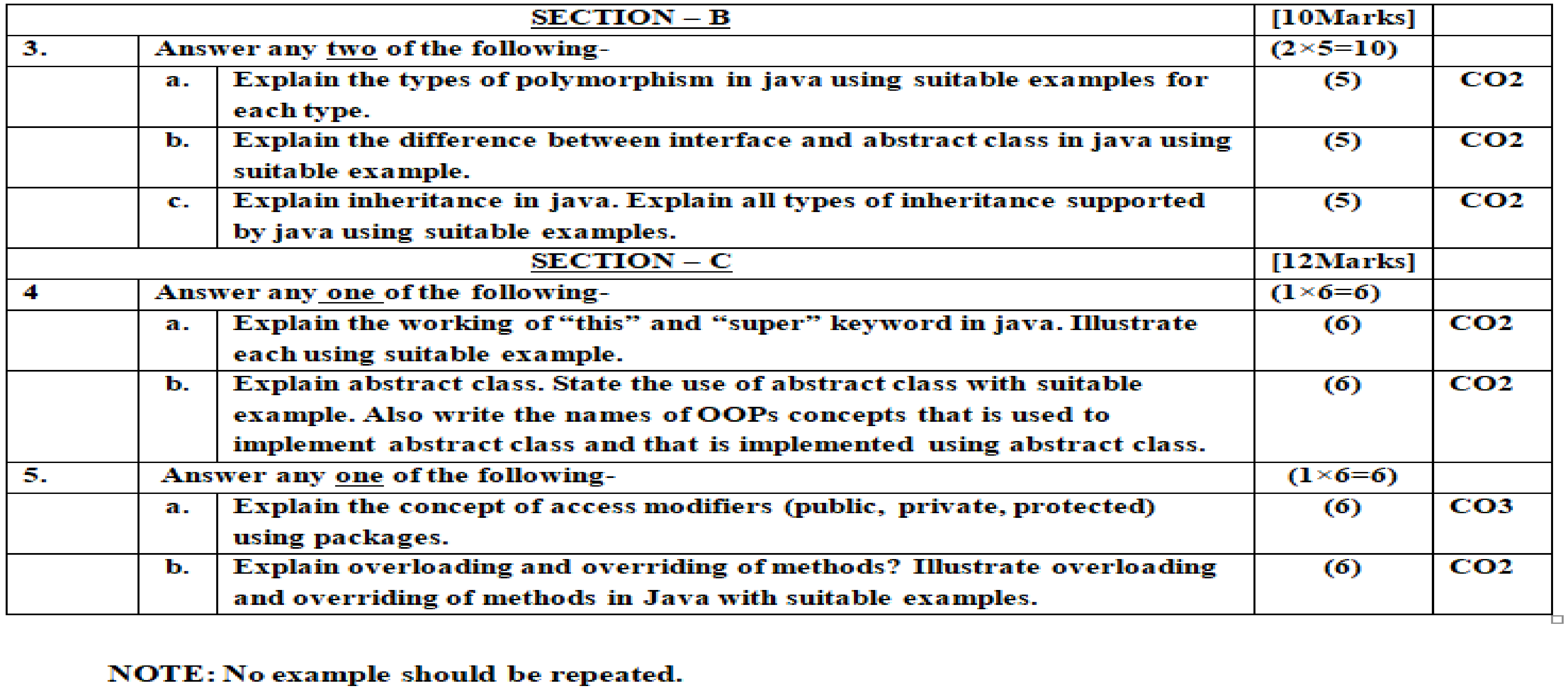


		SECTION – A	[08Marks]	
1.	All questions are compulsory		(4×1=4)	

	a.	Identify the correct way to call the constructor of class “Super” that is inherited by the class “Child”.
		(i) class Child extends Super{ Child(){ Super();}}
		(ii) class Child extends Super{ Child(){ super();}}
		(iii) class Child extends Super{ Child() { super.Super();}}
		(iv) All the ways are correct.
			(1)	CO2

	b.	Total abstraction can be achieved using? (i) abstract class (ii) interface (iii) both (iv) total abstraction cannot be achieved	(1)	CO2
	c.	The .... class inherits all the properties of the ...... class? (i) base, derived (ii) derived base (iii) base, initial (iv) base, final	(1)	CO2
	d.	Runtime polymorphism is also known as: (i) Dynamic binding (ii) Static binding (iii) Early binding (iv) None of the above	(1)	CO2
2.	All questions are compulsory		(2×2=4)	
	a.	Explain a simple program showing garbage collection.	(2)	CO2
	b.	Explain the concept of interface using suitable example.	(2)	CO2

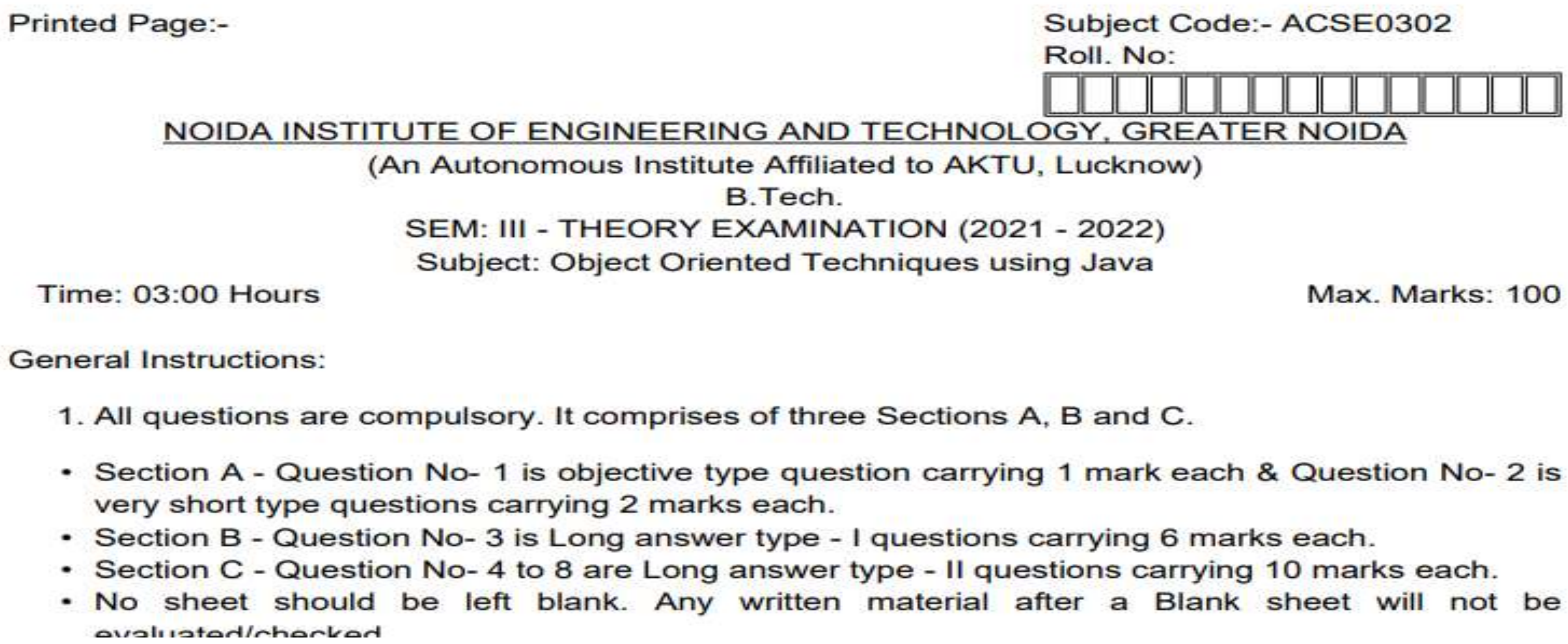
Conti..



		SECTION – B	[10Marks]	
3.	Answer any two of the following-		(2×5=10)	
	a.	Explain the types of polymorphism in java using suitable examples for each type.	(5)	CO2
	b.	Explain the difference between interface and abstract class in java using suitable example.	(5)	CO2
	c.	Explain inheritance in java. Explain all types of inheritance supported by java using suitable examples.	(5)	CO2
		SECTION – C	[12Marks]	
4	Answer any one of the following-		(1×6=6)	
	a.	Explain the working of "this" and "super" keyword in java. Illustrate each using suitable example.	(6)	CO2
	b.	Explain abstract class. State the use of abstract class with suitable example. Also write the names of OOPs concepts that is used to implement abstract class and that is implemented using abstract class.	(6)	CO2
5.	Answer any one of the following-		(1×6=6)	
	a.	Explain the concept of access modifiers (public, private, protected) using packages.	(6)	CO3
	b.	Explain overloading and overriding of methods? Illustrate overloading and overriding of methods in Java with suitable examples.	(6)	CO2

NOTE: No example should be repeated.

Old University Question Paper



Printed Page:-

Subject Code:- ACSE0302
Roll. No:

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA
(An Autonomous Institute Affiliated to AKTU, Lucknow)
B.Tech.
SEM: III - THEORY EXAMINATION (2021 - 2022)
Subject: Object Oriented Techniques using Java

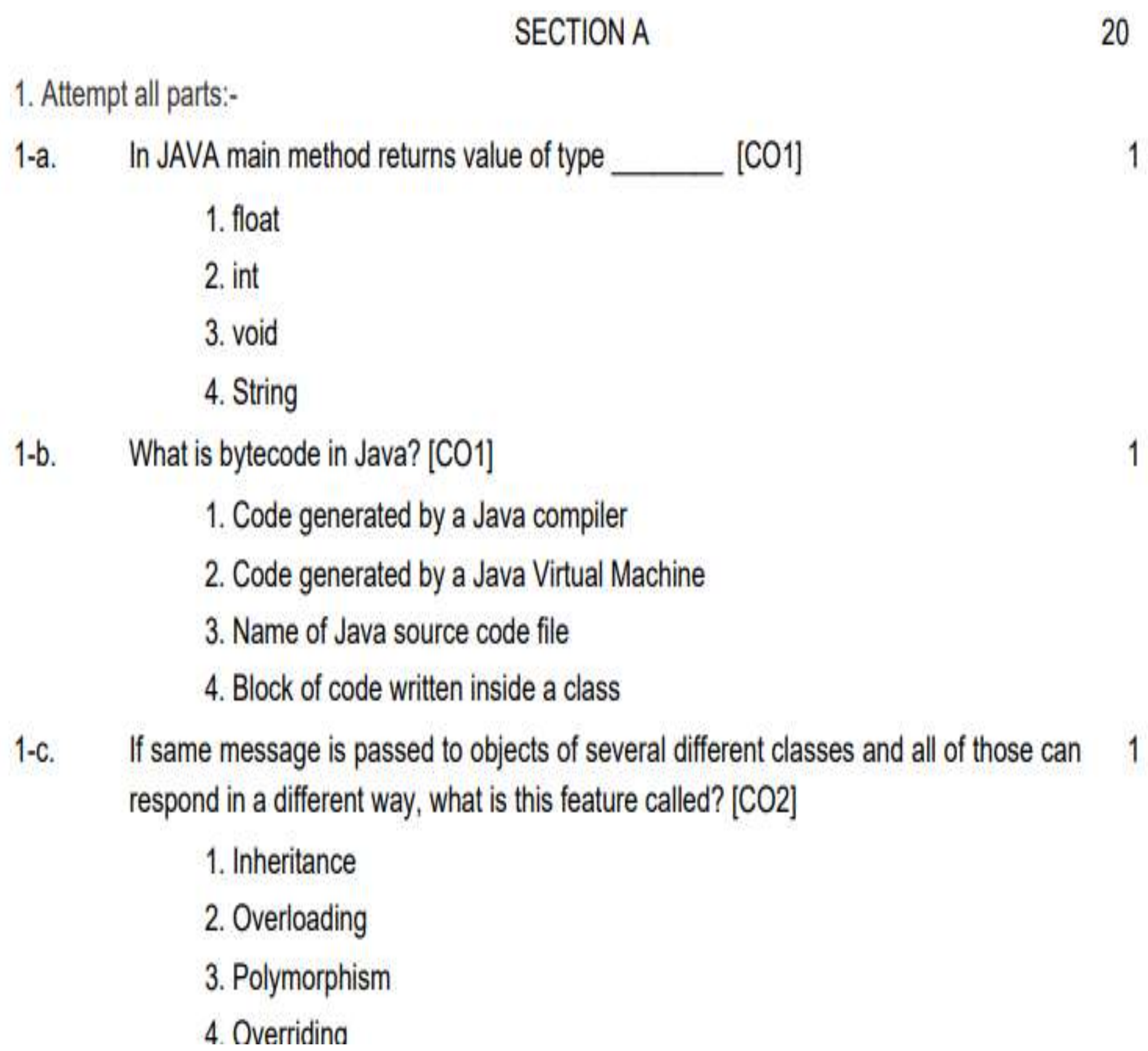
Time: 03:00 Hours Max. Marks: 100

General Instructions:

1. All questions are compulsory. It comprises of three Sections A, B and C.

- Section A - Question No- 1 is objective type question carrying 1 mark each & Question No- 2 is very short type questions carrying 2 marks each.
- Section B - Question No- 3 is Long answer type - I questions carrying 6 marks each.
- Section C - Question No- 4 to 8 are Long answer type - II questions carrying 10 marks each.
- No sheet should be left blank. Any written material after a Blank sheet will not be evaluated/checked

Old University Question Paper



SECTION A 20

1. Attempt all parts:-

1-a. In JAVA main method returns value of type ________ [CO1] 1

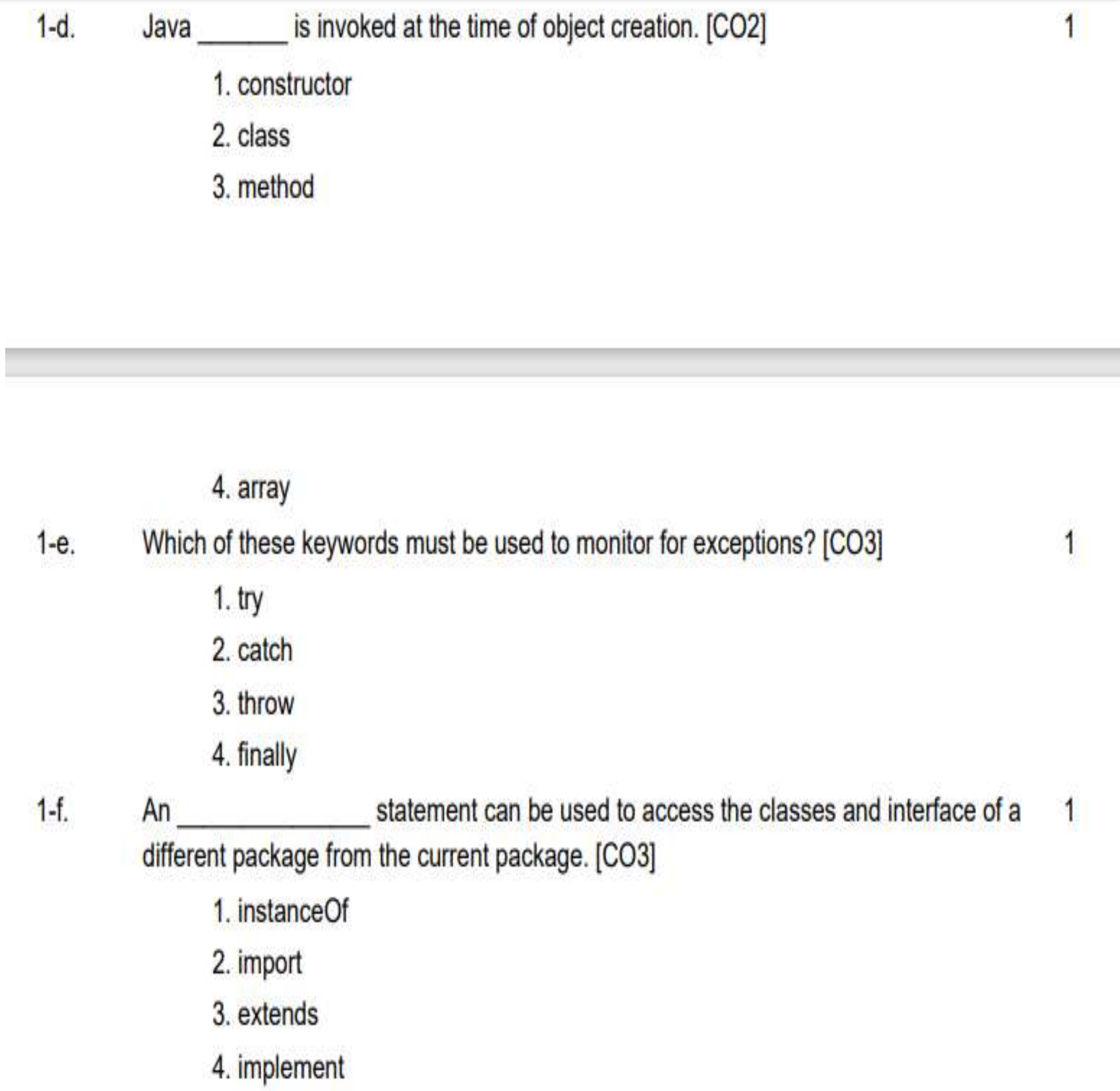
1. float
2. int
3. void
4. String

1-b. What is bytecode in Java? [CO1] 1

1. Code generated by a Java compiler
2. Code generated by a Java Virtual Machine
3. Name of Java source code file
4. Block of code written inside a class

1-c. If same message is passed to objects of several different classes and all of those can respond in a different way, what is this feature called? [CO2] 1

1. Inheritance
2. Overloading
3. Polymorphism
4. Overriding



1-d. Java ______ is invoked at the time of object creation. [CO2] 1

1. constructor
2. class
3. method
4. array

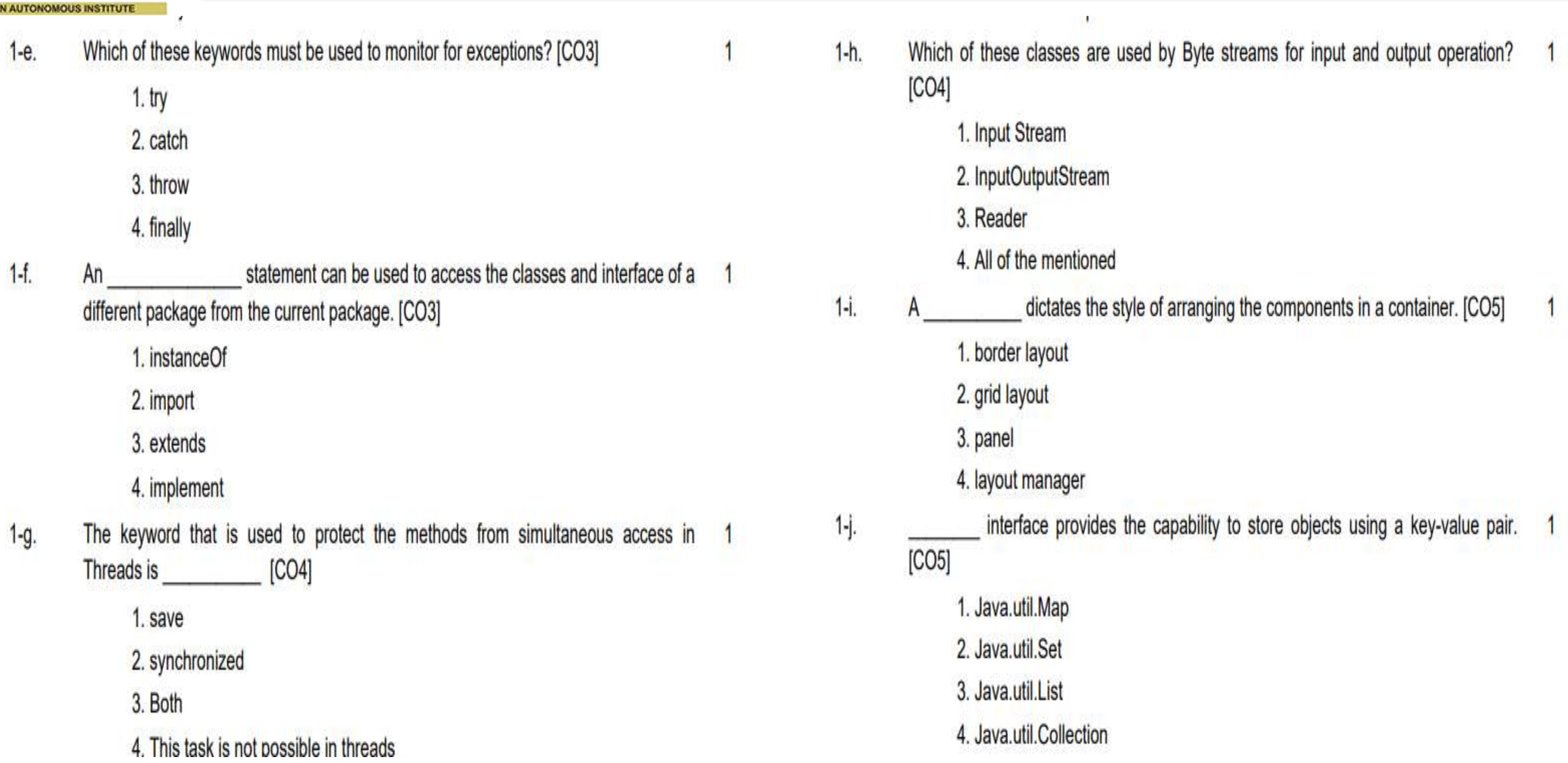
1-e. Which of these keywords must be used to monitor for exceptions? [CO3] 1

1. try
2. catch
3. throw
4. finally

1-f. An ______________ statement can be used to access the classes and interface of a different package from the current package. [CO3] 1

1. instanceOf
2. import
3. extends
4. implement

Old University Question Paper



1-e. Which of these keywords must be used to monitor for exceptions? [CO3] 1

1. try
2. catch
3. throw
4. finally

1-f. An ____________ statement can be used to access the classes and interface of a different package from the current package. [CO3] 1

1. instanceOf
2. import
3. extends
4. implement

1-g. The keyword that is used to protect the methods from simultaneous access in Threads is _________ [CO4] 1

1. save
2. synchronized
3. Both
4. This task is not possible in threads

1-h. Which of these classes are used by Byte streams for input and output operation? [CO4] 1

1. Input Stream
2. InputOutputStream
3. Reader
4. All of the mentioned

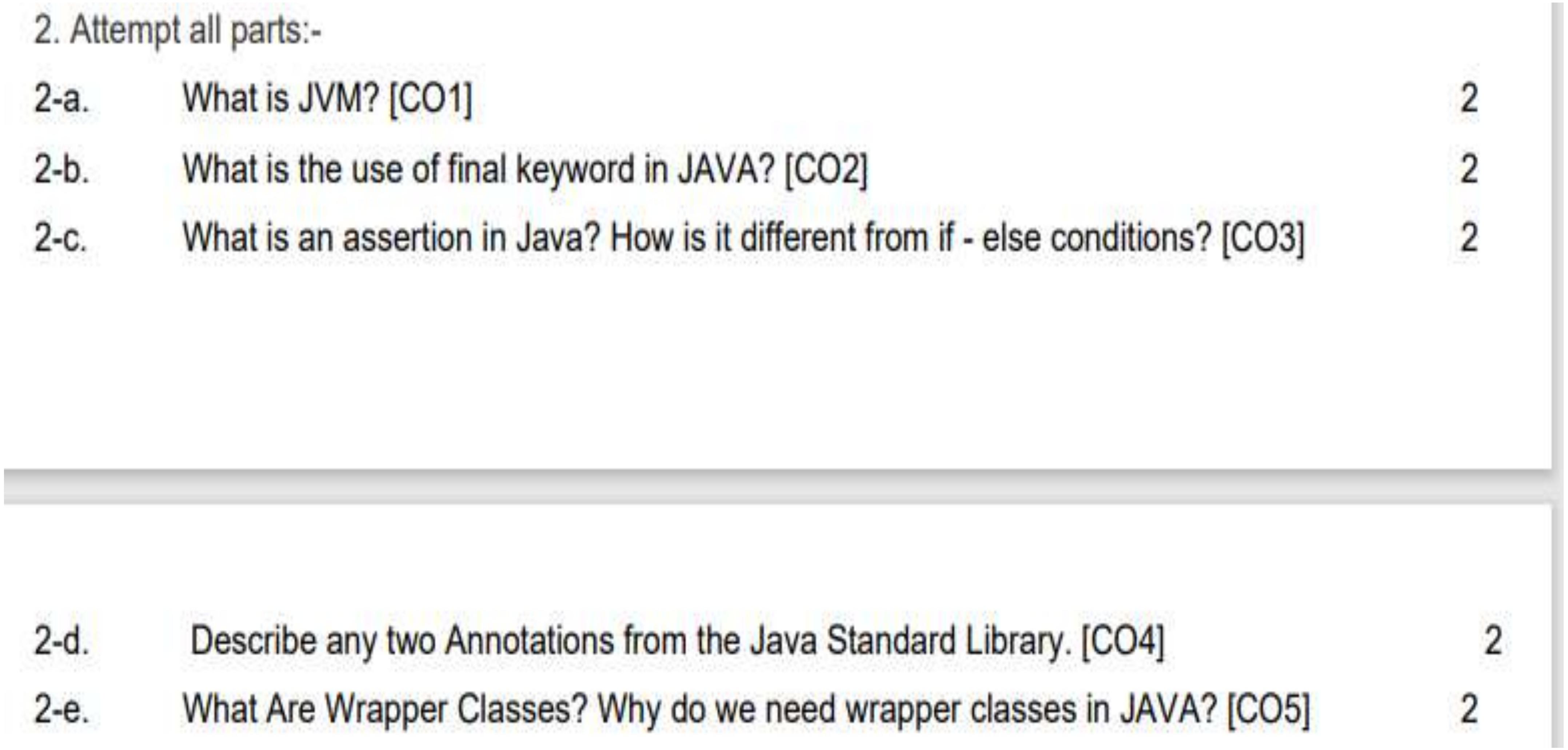
1-i. A _________ dictates the style of arranging the components in a container. [CO5] 1

1. border layout
2. grid layout
3. panel
4. layout manager

1-j. ______ interface provides the capability to store objects using a key-value pair. [CO5] 1

1. Java.util.Map
2. Java.util.Set
3. Java.util.List
4. Java.util.Collection

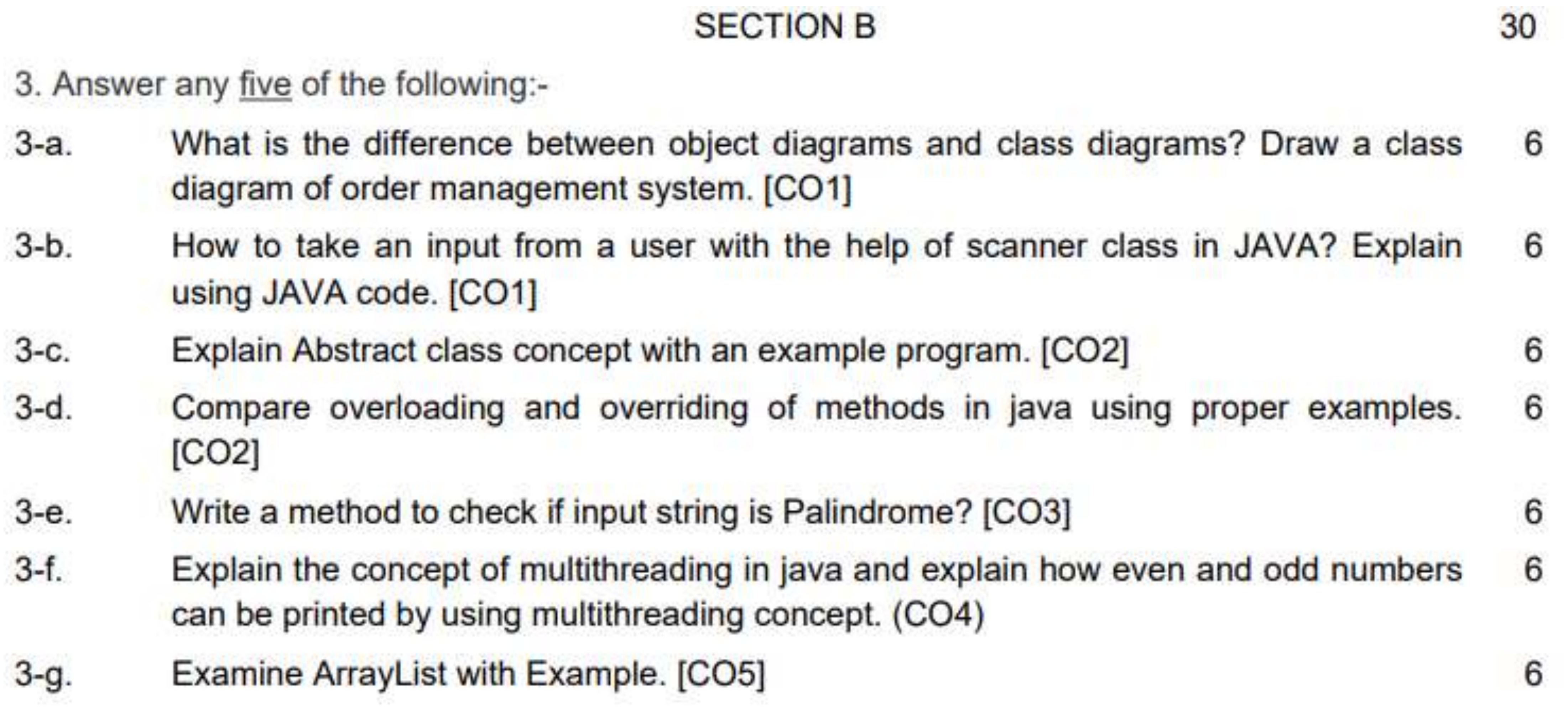
Old University Question Paper



2. Attempt all parts:-

2-a.	What is JVM? [CO1]	2
2-b.	What is the use of final keyword in JAVA? [CO2]	2
2-c.	What is an assertion in Java? How is it different from if - else conditions? [CO3]	2
2-d.	Describe any two Annotations from the Java Standard Library. [CO4]	2
2-e.	What Are Wrapper Classes? Why do we need wrapper classes in JAVA? [CO5]	2

Old University Question Paper

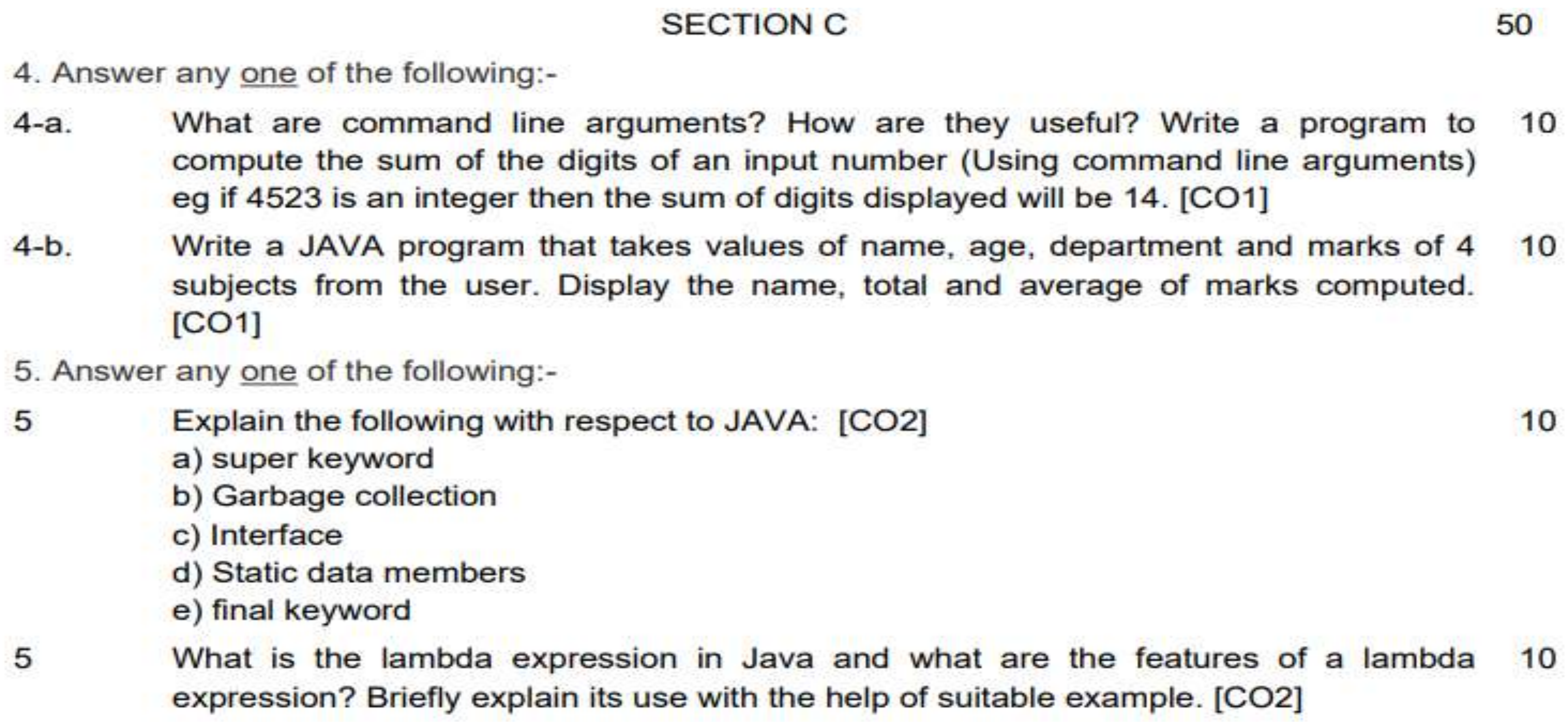


SECTION B 30

3. Answer any five of the following:-

3-a.	What is the difference between object diagrams and class diagrams? Draw a class diagram of order management system. [CO1]	6
3-b.	How to take an input from a user with the help of scanner class in JAVA? Explain using JAVA code. [CO1]	6
3-c.	Explain Abstract class concept with an example program. [CO2]	6
3-d.	Compare overloading and overriding of methods in java using proper examples. [CO2]	6
3-e.	Write a method to check if input string is Palindrome? [CO3]	6
3-f.	Explain the concept of multithreading in java and explain how even and odd numbers can be printed by using multithreading concept. (CO4)	6
3-g.	Examine ArrayList with Example. [CO5]	6

Old University Question Paper



SECTION C 50

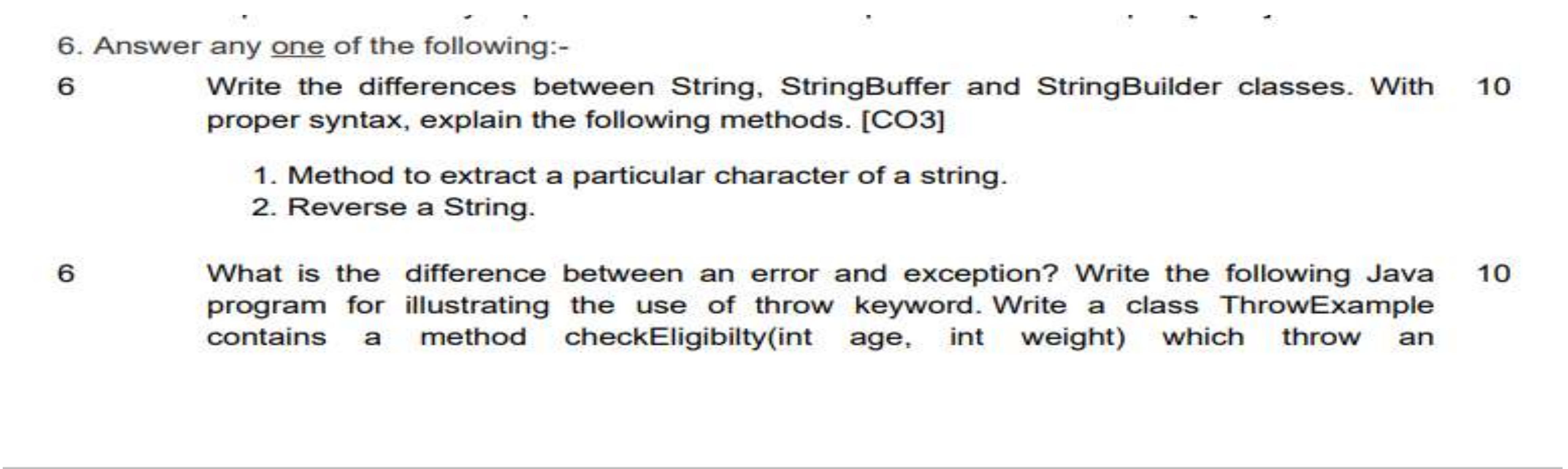
4. Answer any <u>one</u> of the following:-

4-a.	What are command line arguments? How are they useful? Write a program to compute the sum of the digits of an input number (Using command line arguments) eg if 4523 is an integer then the sum of digits displayed will be 14. [CO1]	10
4-b.	Write a JAVA program that takes values of name, age, department and marks of 4 subjects from the user. Display the name, total and average of marks computed. [CO1]	10

5. Answer any <u>one</u> of the following:-

5	Explain the following with respect to JAVA: [CO2] a) super keyword b) Garbage collection c) Interface d) Static data members e) final keyword	10
5	What is the lambda expression in Java and what are the features of a lambda expression? Briefly explain its use with the help of suitable example. [CO2]	10

Old University Question Paper

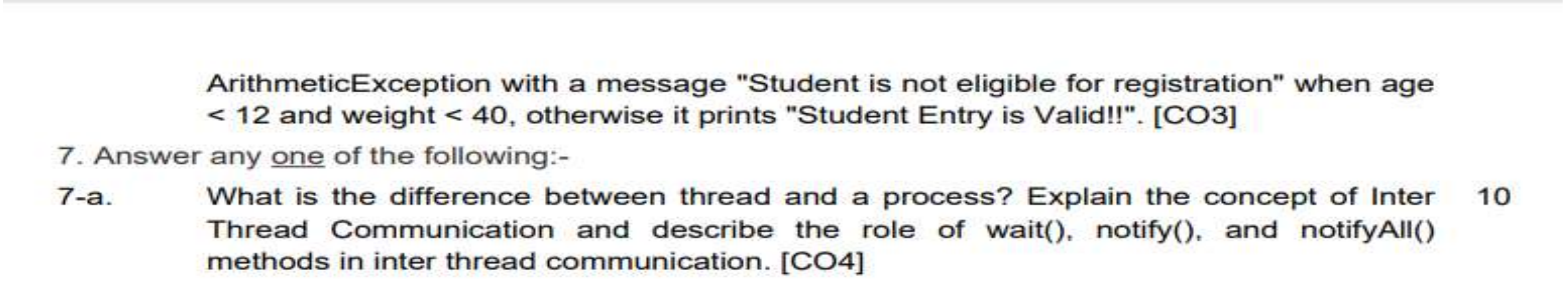


6. Answer any <u>one</u> of the following:-

6 Write the differences between String, StringBuffer and StringBuilder classes. With proper syntax, explain the following methods. [CO3] 10

1. Method to extract a particular character of a string.
2. Reverse a String.

6 What is the difference between an error and exception? Write the following Java program for illustrating the use of throw keyword. Write a class ThrowExample contains a method checkEligibilty(int age, int weight) which throw an 10

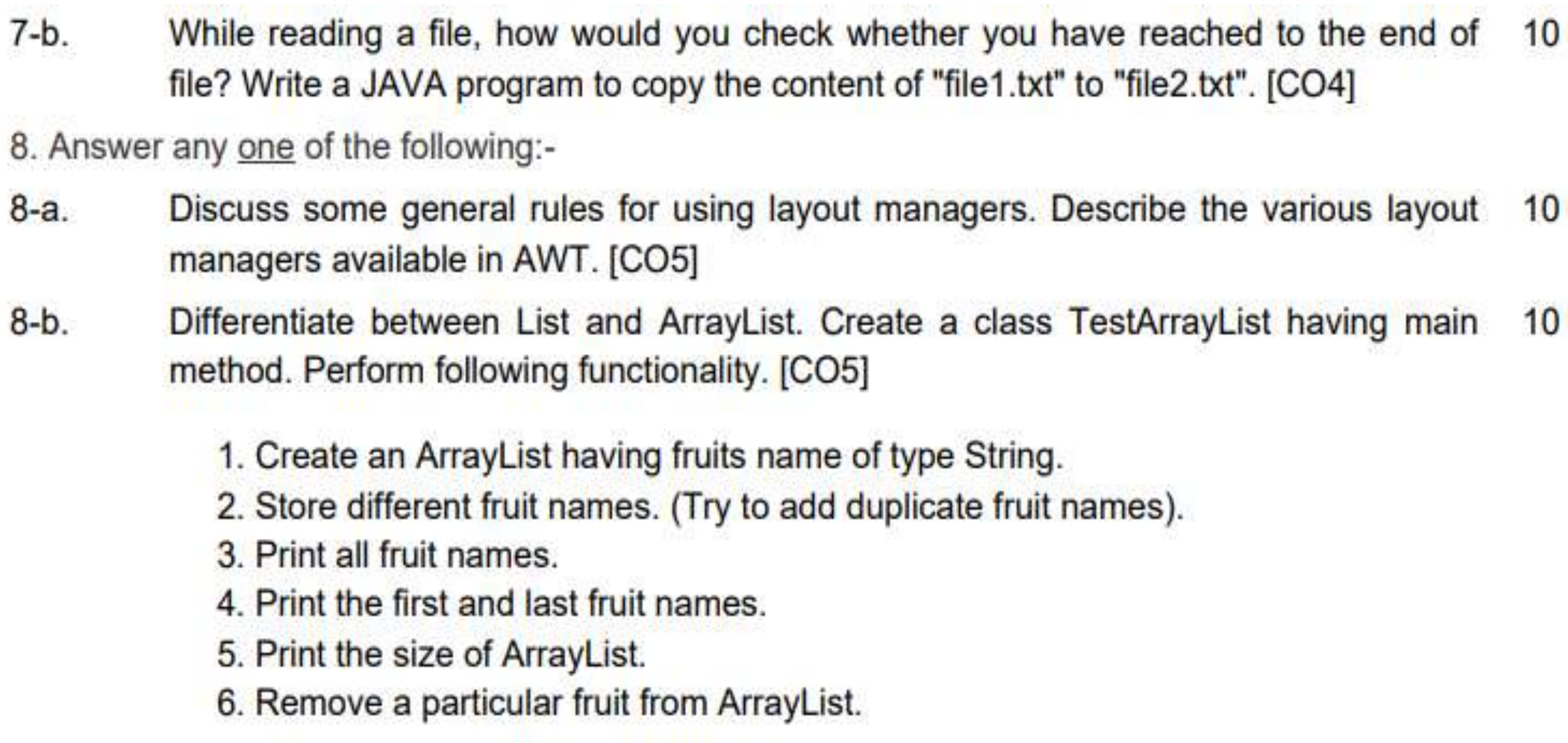


ArithmeticException with a message "Student is not eligible for registration" when age < 12 and weight < 40, otherwise it prints "Student Entry is Valid!!". [CO3]

7. Answer any <u>one</u> of the following:-

7-a. What is the difference between thread and a process? Explain the concept of Inter Thread Communication and describe the role of wait(), notify(), and notifyAll() methods in inter thread communication. [CO4] 10

Old University Question Paper



7-b. While reading a file, how would you check whether you have reached to the end of file? Write a JAVA program to copy the content of "file1.txt" to "file2.txt". [CO4] 10

8. Answer any one of the following:-

8-a. Discuss some general rules for using layout managers. Describe the various layout managers available in AWT. [CO5] 10

8-b. Differentiate between List and ArrayList. Create a class TestArrayList having main method. Perform following functionality. [CO5] 10

1. Create an ArrayList having fruits name of type String.
2. Store different fruit names. (Try to add duplicate fruit names).
3. Print all fruit names.
4. Print the first and last fruit names.
5. Print the size of ArrayList.
6. Remove a particular fruit from ArrayList.

Expected Questions

1. Discuss how to set the priority to threads? What are the different ranges. [CO4]
2. Write a java program to create two threads and execute simultaneously. [CO4]
3. Write a Java program that creates three threads. First thread displays “Hello!” every one second, the second thread displays “Wear Mask !” every two seconds and “Use Sanitizer !” every 5 seconds. [CO4]
4. Write the difference between Extending thread and implementing runnable? [CO4]
5. Explain in detail about thread methods? [CO4]

Old Question Papers

- https://www.iare.ac.in/sites/default/files/IARE_JAVA_MODEL_QP.pdf
- https://www.manaresults.co.in/jntuh/download.php?subcode=133BM

Recap of Unit

- A package is a collection of similar types of Java entities such as classes, interfaces, subclasses, exceptions, errors, and enum.
- In this Unit, we have studied Java exceptions, it's types, and the difference between checked and unchecked exceptions.
- Exception Handling is a mechanism to handle runtime errors such as Class Not Found Exception, IO Exception, SQL Exception, Remote Exception, etc.
- An assertion allows testing the correctness of any assumptions that have been made in the program.
- **String Handling in java**

References

Text Books:
(1) Herbert Schildt," Java - The Complete Reference", McGraw Hill Education 12th edition
(2) Herbert Schildt," Java: A Beginner's Guide", McGraw-Hill Education 2nd edition
(3) James Rumbaugh et. al, "Object Oriented Modeling and Design", PHI 2nd Edition
Reference Books:
(4) Cay S. Horstmann, "Core Java Volume I – Fundamentals", Prentice Hall
(5) Joshua Bloch," Effective Java", Addison Wesley
(6) E Balagurusamy, "Programming with Java A Primer", TMH, 4th edition.

Thank You