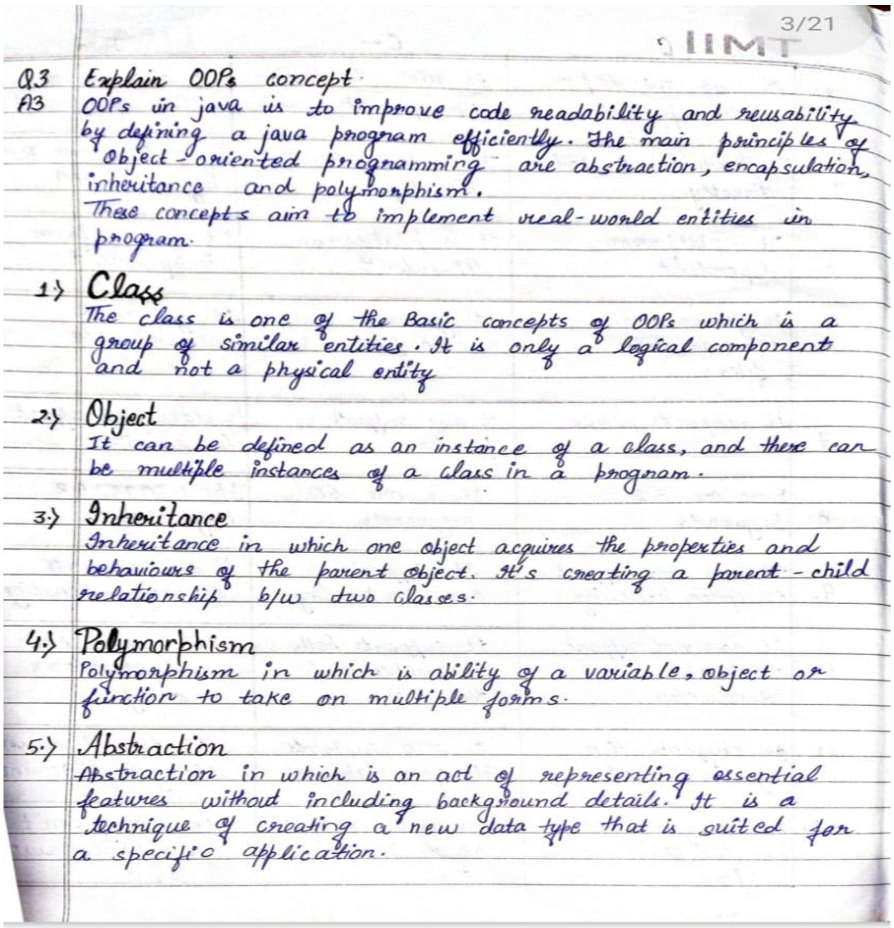
OOPS USING JAVA MATERIAL

Q1. Compare one difference between primitive data type and composite data type with example.

Primitive data types	**Composite data types**
They are predefined/inbuilt data types.	These data types are defined by the user and made-up of primitive data type values.
Examples: int, byte, long, short, char, float, double, boolean.	Examples: Array, class, interface.

Q2. Define Object Oriented Programming and contrast four characteristics/principal of Oop's.


3/21
IIMT

Q3 Explain OOPs concept.
A3 OOPs in java is to improve code readability and reusability by defining a java program efficiently. The main principles of object-oriented programming are abstraction, encapsulation, inheritance and polymorphism.
These concepts aim to implement real-world entities in program.

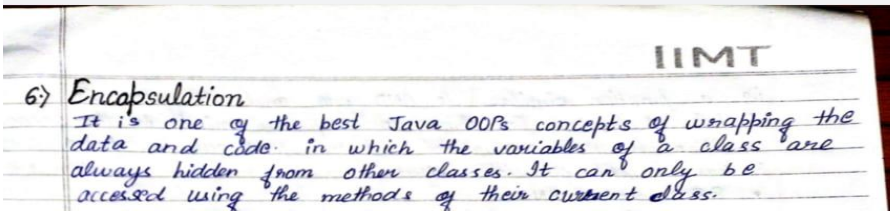
1.) Class
The class is one of the Basic concepts of OOPs which is a group of similar entities. It is only a logical component and not a physical entity

2.) Object
It can be defined as an instance of a class, and there can be multiple instances of a class in a program.

3.) Inheritance
Inheritance in which one object acquires the properties and behaviours of the parent object. It's creating a parent - child relationship b/w two classes.

4.) Polymorphism
Polymorphism in which is ability of a variable, object or function to take on multiple forms.

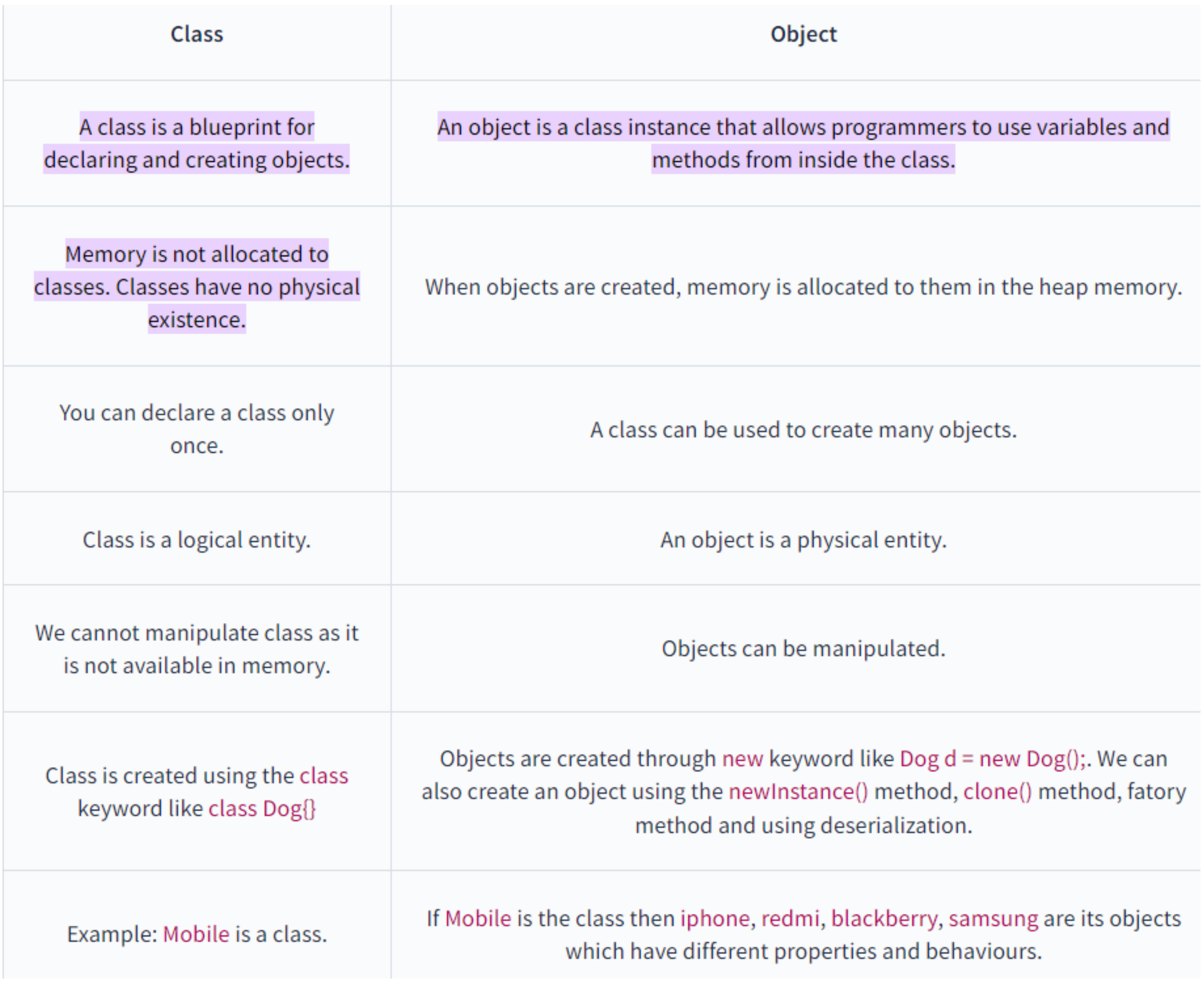
5.) Abstraction
Abstraction in which is an act of representing essential features without including background details. It is a technique of creating a new data type that is suited for a specific application.


IIMT

6.) Encapsulation
It is one of the best Java OOPs concepts of wrapping the data and code. in which the variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class.

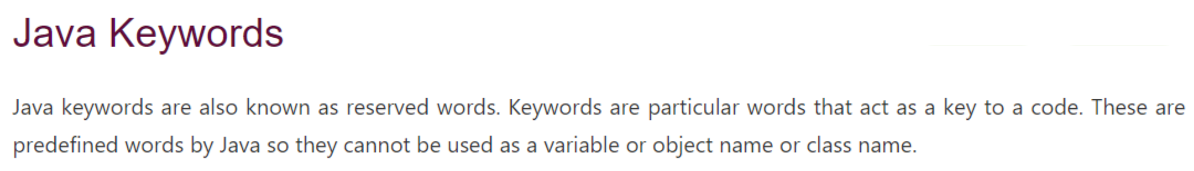
OOPS USING JAVA MATERIAL

Q3. Identify two differences between a class and an object.



Class	Object
A class is a blueprint for declaring and creating objects.	An object is a class instance that allows programmers to use variables and methods from inside the class.
Memory is not allocated to classes. Classes have no physical existence.	When objects are created, memory is allocated to them in the heap memory.
You can declare a class only once.	A class can be used to create many objects.
Class is a logical entity.	An object is a physical entity.
We cannot manipulate class as it is not available in memory.	Objects can be manipulated.
Class is created using the class keyword like class Dog{}	Objects are created through new keyword like Dog d = new Dog();. We can also create an object using the newInstance() method, clone() method, fatory method and using deserialization.
Example: Mobile is a class.	If Mobile is the class then iphone, redmi, blackberry, samsung are its objects which have different properties and behaviours.

Q4. Tell, what are keywords? Give an example.

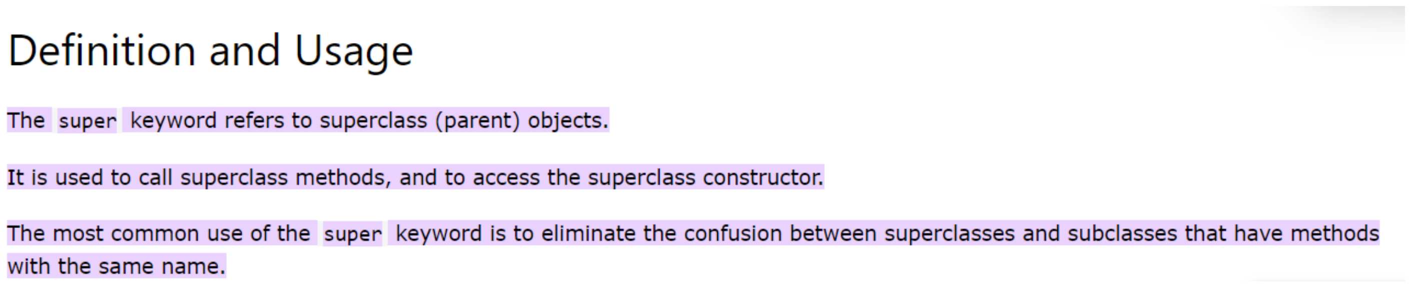


Java Keywords

Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

Example: abstract, boolen, break, class etc.

Q5. State the use of “super” keyword in inheritance.



Definition and Usage

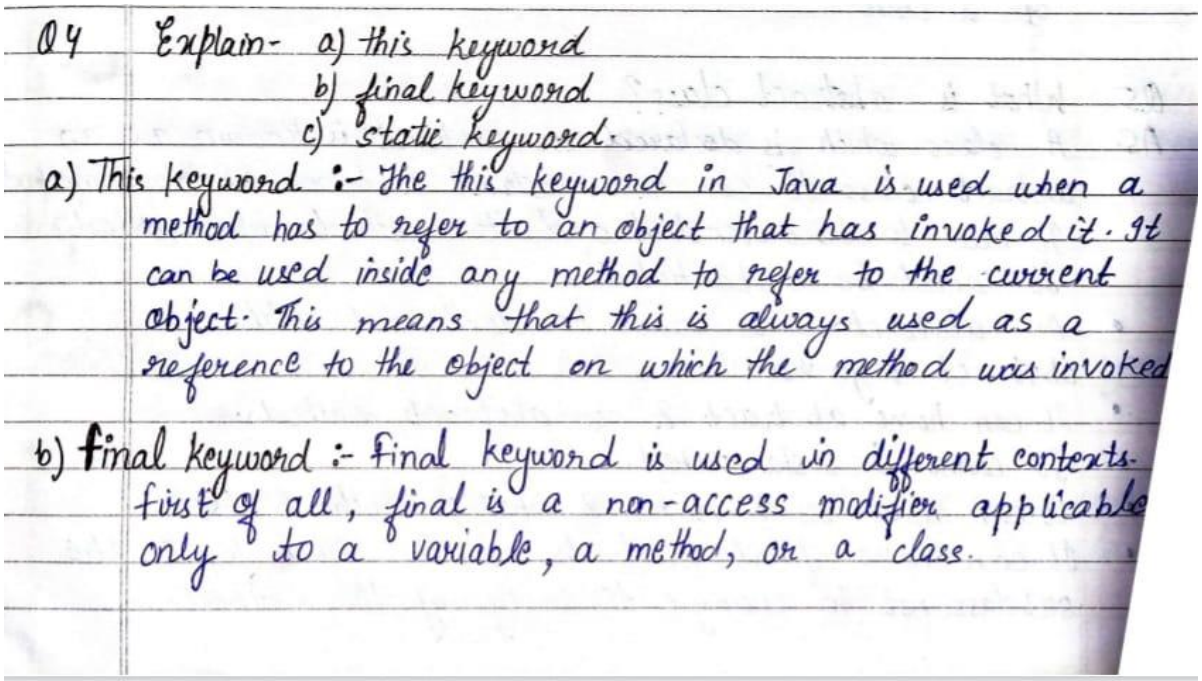
The `super` keyword refers to superclass (parent) objects.

It is used to call superclass methods, and to access the superclass constructor.

The most common use of the `super` keyword is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

OOPS USING JAVA MATERIAL

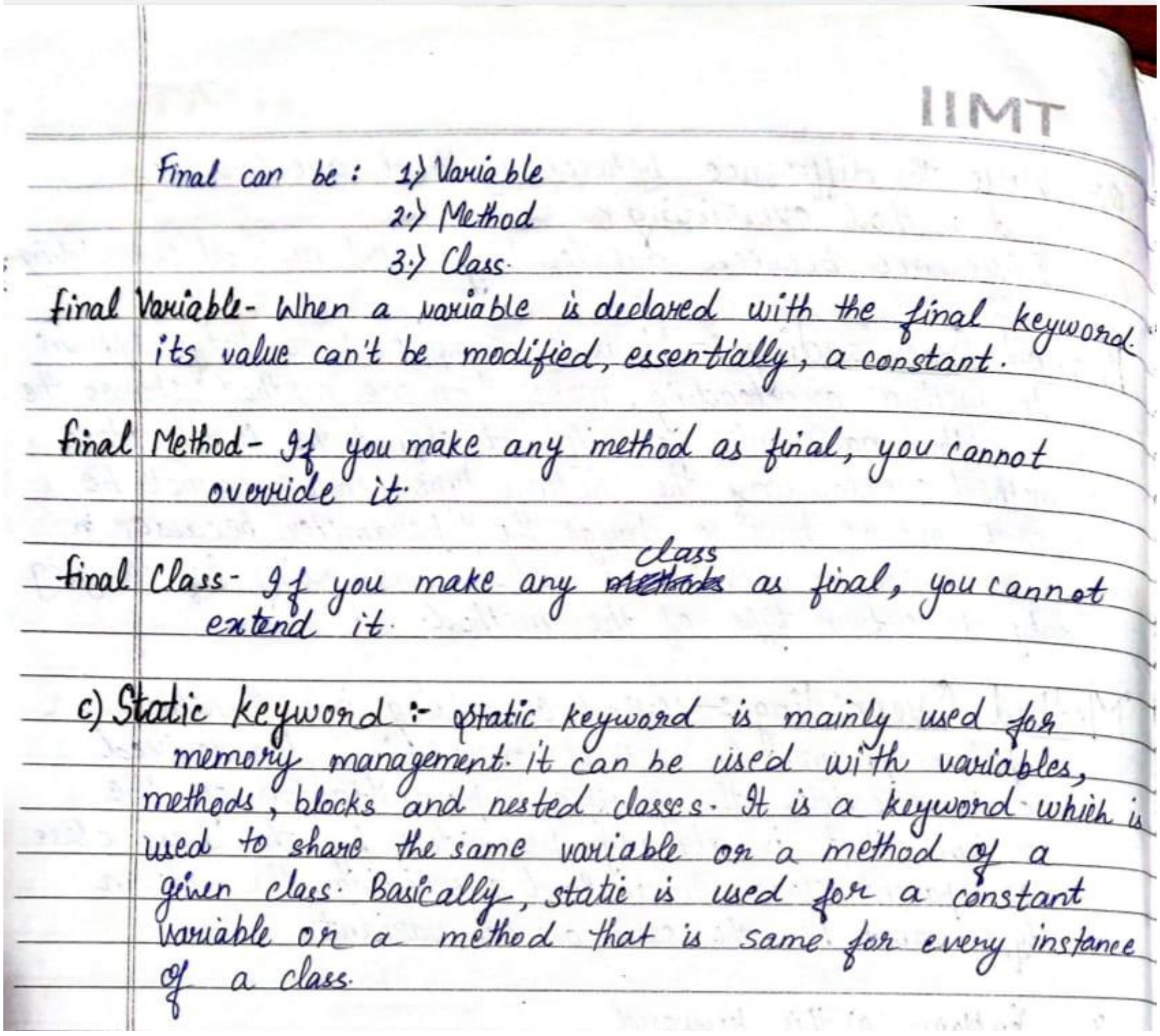
Q6. Explain (a)this keyword, (b)final keyword, (c)static keyword.



Q4 Explain- a) this keyword
b) final keyword
c) static keyword.

a) This keyword :- The this keyword in Java is used when a method has to refer to an object that has invoked it. It can be used inside any method to refer to the current object. This means that this is always used as a reference to the object on which the method was invoked

b) final keyword :- Final keyword is used in different contexts. first of all, final is a non-access modifier applicable only to a variable, a method, or a class.



IIMT

Final can be : 1.) Variable
2.) Method
3.) Class.

final Variable- When a variable is declared with the final keyword its value can't be modified, essentially, a constant.

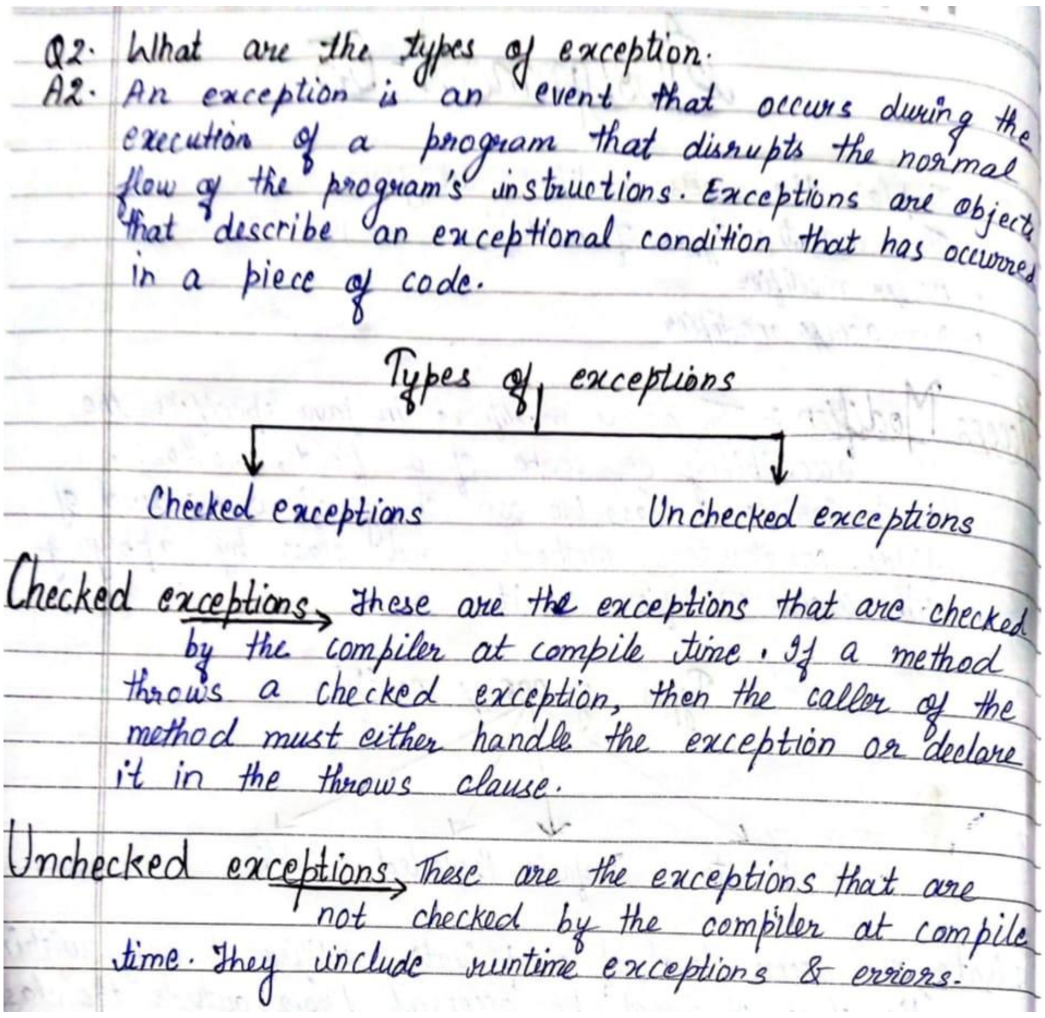
final Method- If you make any method as final, you cannot override it.

final Class- If you make any ~~methods~~ class as final, you cannot extend it.

c) Static keyword :- static keyword is mainly used for memory management. it can be used with variables, methods, blocks and nested classes. It is a keyword which is used to share the same variable or a method of a given class. Basically, static is used for a constant variable or a method that is same for every instance of a class.

OOPS USING JAVA MATERIAL

Q.7 What are the types of exception.



Q2. What are the types of exception.
A2. An exception is an event that occurs during the execution of a program that disrupts the normal flow of the program's instructions. Exceptions are object that describe an exceptional condition that has occurred in a piece of code.

Types of exceptions

Checked exceptions | Unchecked exceptions

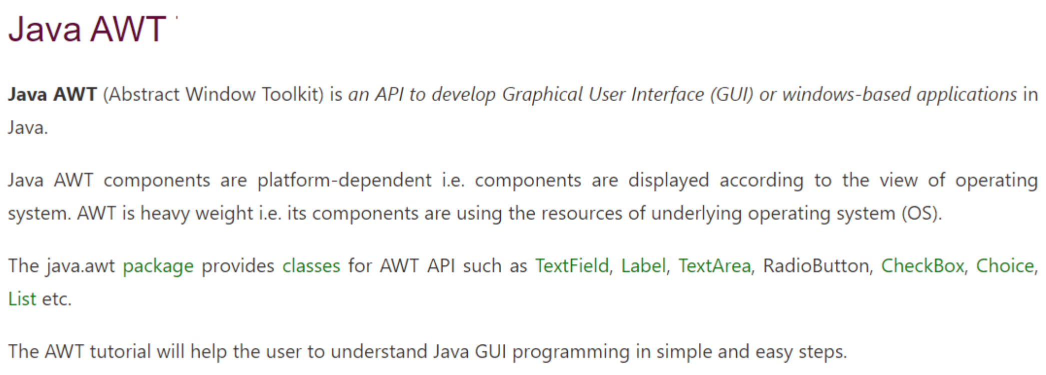
Checked exceptions → These are the exceptions that are checked by the compiler at compile time. If a method throws a checked exception, then the caller of the method must either handle the exception or declare it in the throws clause.

Unchecked exceptions → These are the exceptions that are not checked by the compiler at compile time. They include runtime exceptions & errors.

Q8. Evaluate, how does an exception propagate in the code?

Exception propagation in Java occurs when an exception thrown from the top of the stack. When it is not caught, the exception drops down the call stack of the preceding method. If it is not caught there, it further drops down to the previous method. This continues until the method reaches the bottom of the call stack or is caught somewhere in between.

Q9. Explain AWT in java.



Java AWT

Java AWT (Abstract Window Toolkit) is *an API to develop Graphical User Interface (GUI) or windows-based applications* in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps.

Q10. What is RDBMS.

RDBMS stands for Relational Database Management System. It is an information management system that is oriented on a data model. Here all the information is properly stored as tables. RDBMS Example systems are SQL Server, Oracle, MySQL, MariaDB, and SQLite.

Q11. Contrast on JIT compiler in java.

A just-in-time (JIT) compiler is a program that turns bytecode into instructions that can be sent directly to a computer's processor (CPU). Typically, compilers are key in deciding the speed of an application for developers and end users. Just-in-time compilers can be used for performance optimization to improve application runtime.

JIT compilers contrast different compiler types such as a traditional compiler, which will compile all code to a machine language *before* a program starts to run. Newer programs will make use of JIT compilers, which generate code while the program is running.

Q12. illustrate the difference between this () and super () in Java.

	super()	this()
Definition	super() - refers immediate parent class instance.	this() - refers current class instance.
Invoke	Can be used to invoke immediate parent class method.	Can be used to invoke current class method.
Constructor	super() acts as immediate parent class constructor and should be first line in child class constructor.	this() acts as current class constructor and can be used in parametrized constructors.
Override	When invoking a superclass version of an overridden method the super keyword is used.	When invoking a current version of an overridden method the this keyword is used.

Q13. What is Constructor and its role and types.

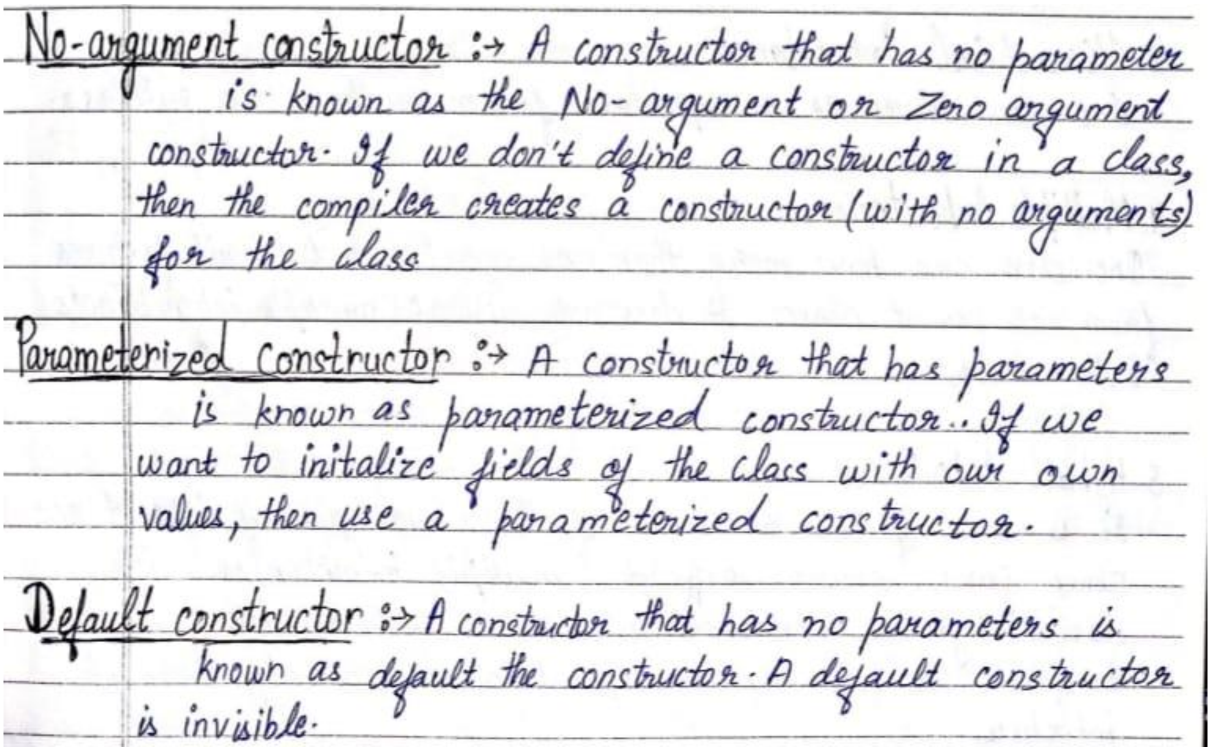
A constructor is a special function that creates and initializes an object instance of a class. In JavaScript, a constructor gets called when an object is created using the `new` keyword.

The purpose of a constructor is to create a new object and set values for any existing object properties.

When a constructor gets invoked in JavaScript, the following sequence of operations take place:

- A new empty object gets created.
- The `this` keyword begins to refer to the new object and it becomes the current instance object.
- The new object is then returned as the return value of the constructor.

Types:



No-argument constructor :→ A constructor that has no parameter is known as the No-argument or Zero argument constructor. If we don't define a constructor in a class, then the compiler creates a constructor (with no arguments) for the class

Parameterized Constructor :→ A constructor that has parameters is known as parameterized constructor. If we want to initalize fields of the class with our own values, then use a parameterized constructor.

Default constructor :→ A constructor that has no parameters is known as default the constructor. A default constructor is invisible.

Q14. Can you call a constructor of a class inside another constructor? (Constructor chaining).

In constructor chain, a constructor is called from another constructor in the same class this process is known as **constructor chaining.** It occurs through inheritance. When we create an instance of a derived class, all the constructors of the inherited class (base class) are first invoked, after that the constructor of the calling class (derived class) is invoked.

We can achieve constructor chaining in two ways:

- **Within the same class:** If the constructors belong to the same class, we use **this**
- **From the base class:** If the constructor belongs to different classes (parent and child classes), we use the **super** keyword to call the constructor from the base class.

OOPS USING JAVA MATERIAL

Q15. Define parameterized constructors in Java.

Parameterized constructors

A parameterized constructor accepts parameters with which you can initialize the instance variables. Using parameterized constructor, you can initialize the class variables dynamically at the time of instantiating the class with distinct values.

Syntax

```
public class Sample{
   Int i;
   public sample(int i){
      this.i = i;
   }
}
```

Example

Live Demo

```
public class Test {
   String val;
   Test(String val){
      this.val = val;
   }
   public static void main(String args[]){
      Test obj = new Test("test");
      System.out.println(obj.val);
   }
}
```

Output

```
test
```

OOPS USING JAVA MATERIAL

Q16. Define package in Java. List down various advantages of packages.

Java Package

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Here, we will have the detailed learning of creating and using user-defined packages.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

Q17. Evaluate, how does an exception propagate in the code?

Exception propagation in Java occurs when an exception thrown from the top of the stack. When it is not caught, the exception drops down the call stack of the preceding method. If it is not caught there, it further drops down to the previous method. This continues until the method reaches the bottom of the call stack or is caught somewhere in between.

Example

Let us see an example which illustrates exception propagation in Java –

Live Demo

```java
public class Example {
   void method1() // generates an exception {
      int arr[] = {10,20,30};
      System.out.println(arr[7]);
   }
   void method2() // doesn't catch the exception {
      method1();
   }
   // method1 drops down the call stack
   void method3() // method3 catches the exception {
      try {
         method2();
      } catch(ArrayIndexOutOfBoundsException ae) {
         System.out.println("Exception is caught");
      }
   }
   public static void main(String args[]) {
      Example obj = new Example();
      obj.method3();
   }
}
```

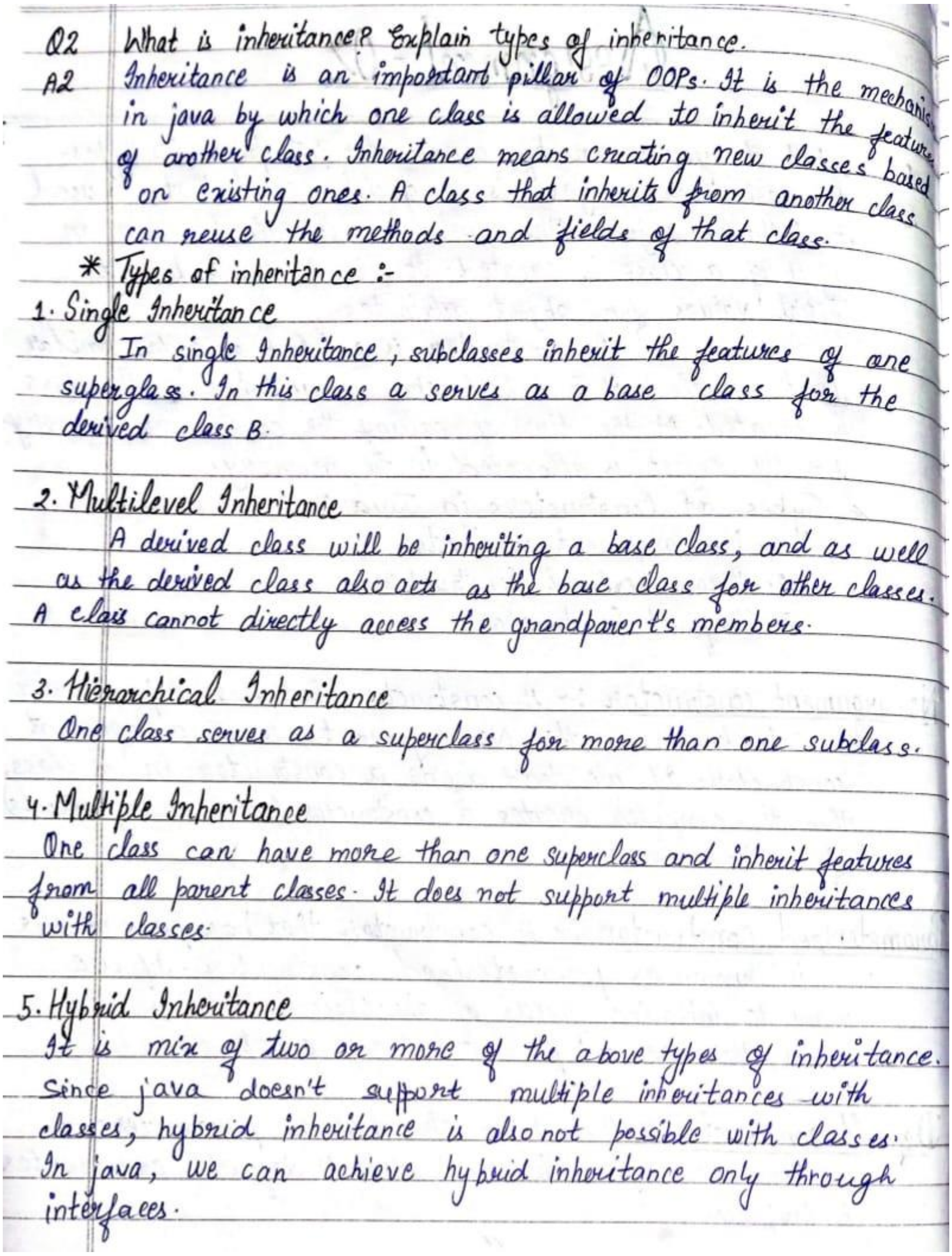
Output

The output is as follows –

```
Exception is caught
```

OOPS USING JAVA MATERIAL

Q18. What is inheritance? Explain types of inheritance.



Q2 What is inheritance? Explain types of inheritance.

A2 Inheritance is an important pillar of OOPs. It is the mechanis in java by which one class is allowed to inherit the feature of another class. Inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class.

* Types of inheritance :-

1. Single Inheritance

In single Inheritance, subclasses inherit the features of one superglass. In this class a serves as a base class for the derived class B.

2. Multilevel Inheritance

A derived class will be inheriting a base class, and as well as the derived class also acts as the base class for other classes. A class cannot directly access the grandparent's members.

3. Hierarchical Inheritance

One class serves as a superclass for more than one subclass.

4. Multiple Inheritance

One class can have more than one superclass and inherit features from all parent classes. It does not support multiple inheritances with classes.

5. Hybrid Inheritance

It is mix of two or more of the above types of inheritance. Since java doesn't support multiple inheritances with classes, hybrid inheritance is also not possible with classes. In java, we can achieve hybrid inheritance only through interfaces.

OOPS USING JAVA MATERIAL

Q19. Will the finally block get executed when the return statement is written at the end of try block and catch block?

Yes, the finally block will be executed even after a return statement in a method.

The **finally block** will always execute even an exception occurred or not in Java. If we call the **System.exit()** method explicitly in the **finally block** then only it will not be executed. There are few situations where the finally will not be executed like **JVM crash**, **power failure**, **software crash** and etc. Other than these conditions, the **finally block** will be always executed.

Example

```
public class FinallyBlockAfterReturnTest {
   public static void main(String[] args) {
      System.out.println(count());
   }
   public static int count() {
      try {
         return 1;
      } catch(Exception e) {
         return 2;
      } finally {
         System.out.println("Finally block will execute even after a return statement in a metho
      }
   }
}
```

Output

```
Finally block will always excute even after a return statement in a method
1
```

Q20. Define untrusted applets.

Untrusted applets are those Java applets that cannot access or execute local system files. By default, all downloaded applets are considered as untrusted.

Q21. Summarize, what are the restrictions imposed on Java applets?

Mostly due to security reasons, the following restrictions are imposed on Java applets:

- An applet cannot load libraries or define native methods.
- An applet cannot ordinarily read or write files on the execution host.
- An applet cannot read certain system properties.
- An applet cannot make network connections except to the host that it came from.
- An applet cannot start any program on the host that's executing it.

OOPS USING JAVA MATERIAL

Q22. Define, what is the difference between applets loaded over the internet and applets loaded via the file system?

Regarding the case where an applet is loaded over the internet, the applet is loaded by the applet classloader and is subject to the restrictions enforced by the applet security manager. Regarding the case where an applet is loaded from the client's local disk, the applet is loaded by the file system loader. Applets loaded via the file system are allowed to read files, write files and to load libraries on the client. Also, applets loaded via the file system are allowed to execute processes and finally, applets loaded via the file system are not passed through the byte code verifier.

Q23. What is JDBC. Explain the role of Driver in JDBC.

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.

A JDBC driver (Java Database Connectivity driver) is a small piece of software that allows JDBC to connect to different databases.

Essentially, a JDBC driver makes it possible to do three things:

1. Establish a connection with a data source.
2. Send queries and update statements to the data source.
3. Process the results.

Q24. Discover the use of Callable Statement in JDBC.

CallableStatement interface is used to call the **stored procedures and functions**.

We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.

Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.

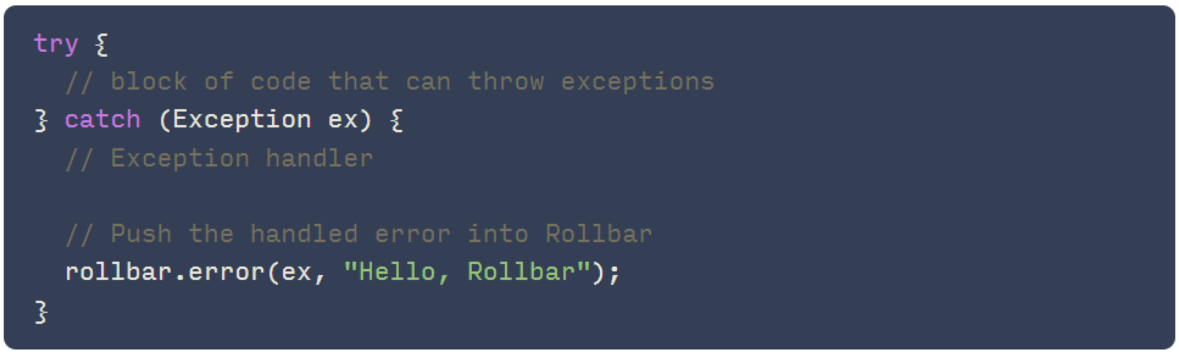
Q25. How can you handle Java exceptions? Relate with an example in Java.

How to handle exceptions in Java

Let's start with the basics of exception handling in Java before we move to more advanced topics. The `try-catch` is the simplest method of handling exceptions. Put the code you want to run in the `try` block, and any Java exceptions that the code throws are caught by one or more `catch` blocks.

This method will catch any type of Java exceptions that get thrown. This is the simplest mechanism for handling exceptions.

Example:

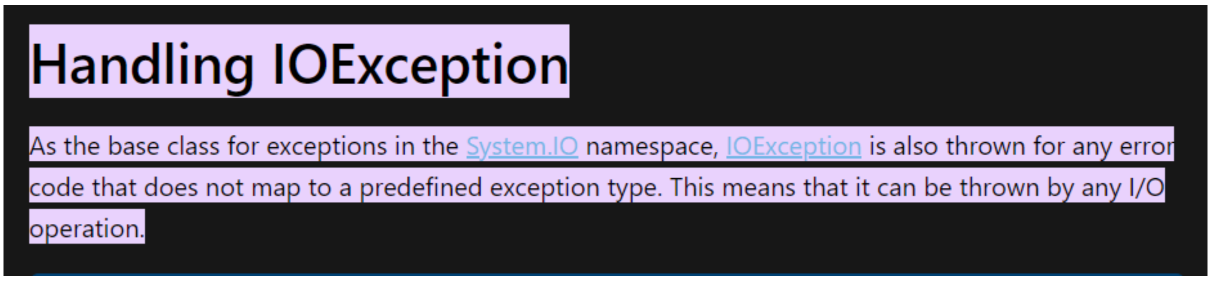


```
try {
  // block of code that can throw exceptions
} catch (Exception ex) {
  // Exception handler

  // Push the handled error into Rollbar
  rollbar.error(ex, "Hello, Rollbar");
}
```

Note: You can't use a `try` block alone. The try block should be immediately followed either by a `catch` or `finally` block.

Q26. Contrast, how I/O Exception helpful in handling I/O errors?



Handling IOException

As the base class for exceptions in the System.IO namespace, IOException is also thrown for any error code that does not map to a predefined exception type. This means that it can be thrown by any I/O operation.

Q27. Explain exception hierarchy in java.

The hierarchy of Exceptions in the Java programming language begins with the Throwable class – which comes from the Object class and is its direct subclasswhileThe Exception class presents all This Throwable class further branches into two subclasses – Error and Exception. Here's a flowchart to understand the Java Exception hierarchy better:

Q28. Define applet in Java.

Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Advantage of Applet

There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Q29. State the life cycle of Applet programming.

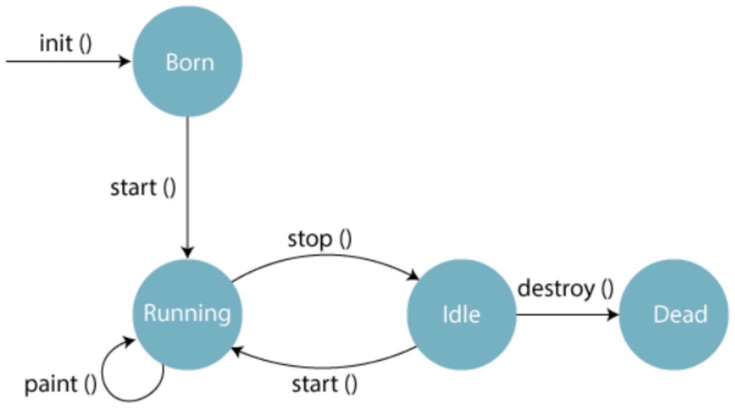
Applet Life Cycle in Java

In Java, an applet is a special type of program embedded in the web page to generate dynamic content. Applet is a class in Java.

The applet life cycle can be defined as the process of how the object is created, started, stopped, and destroyed during the entire execution of its application. It basically has five core methods namely init(), start(), stop(), paint() and destroy().These methods are invoked by the browser to execute.

Along with the browser, the applet also works on the client side, thus having less processing time.

Methods of Applet Life Cycle



Q30. What is the applet class loader, and what does it provide ?

When an applet is loaded over the internet, the applet is loaded by the applet classloader. The class loader enforces the Java name space hierarchy. Also, the class loader guarantees that a unique namespace exists for classes that come from the local file system, and that a unique namespace exists for each network source. When a browser loads an applet over the net, that applet's classes are placed in a private namespace associated with the applet's origin. Then, those classes loaded by the class loader are passed through the verifier.The verifier checks that the class file conforms to the Java language specification . Among other things, the verifier ensures that there are no stack overflows or underflows and that the parameters to all bytecode instructions are correct.

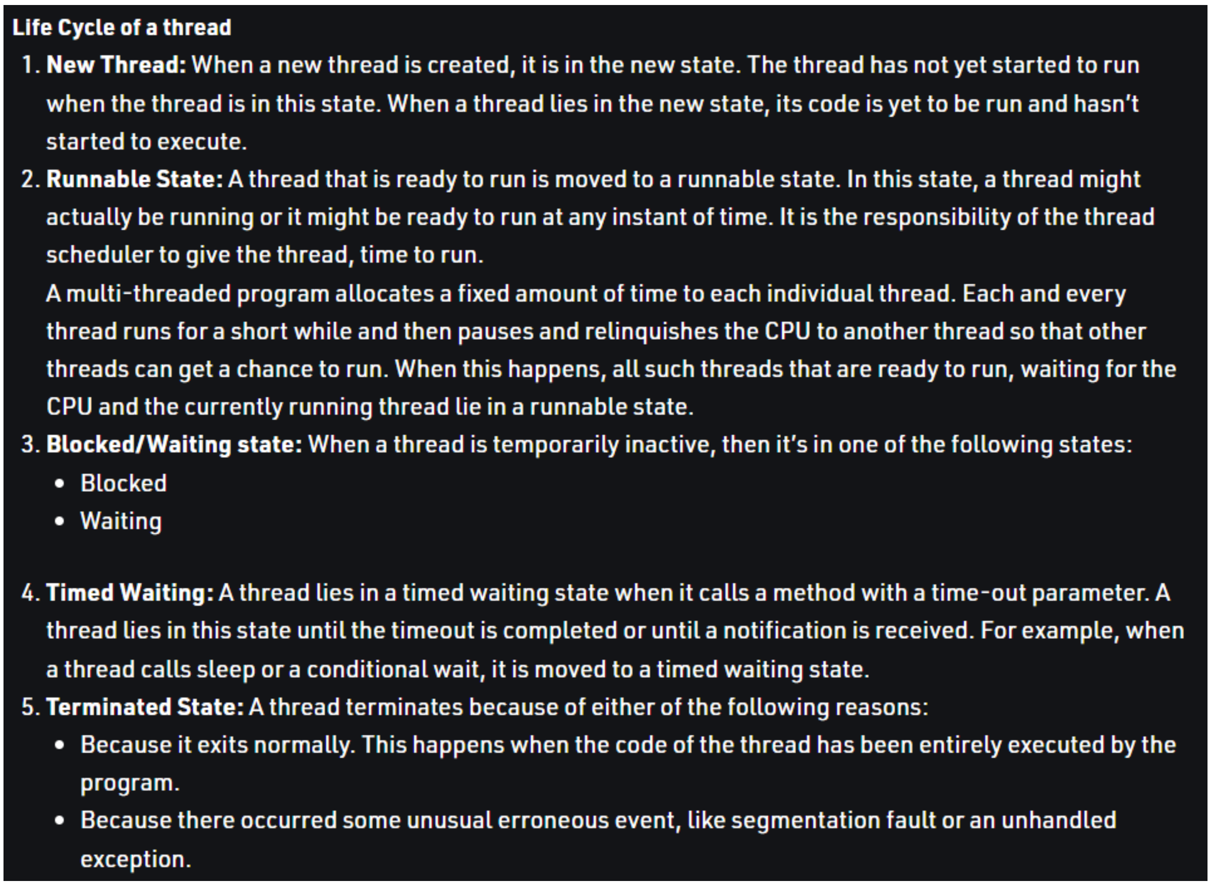
Q31. Recognize the advantages of multithreading.

Multithreading allows the execution of multiple parts of a program at the same time. These parts are known as threads and are lightweight processes available within the process. So multithreading leads to maximum utilization of the CPU by multitasking.

Advantages:

- **Resource Sharing**
  All the threads of a process share its resources such as memory, data, files etc. A single application can have different threads within the same address space using resource sharing.
- **Responsiveness**
  Program responsiveness allows a program to run even if part of it is blocked using multithreading. This can also be done if the process is performing a lengthy operation. For example - A web browser with multithreading can use one thread for user contact and another for image loading at the same time.
- **Utilization of Multiprocessor Architecture**
  In a multiprocessor architecture, each thread can run on a different processor in parallel using multithreading. This increases concurrency of the system. This is in direct contrast to a single processor system, where only one process or thread can run on a processor at a time.
- **Economy**
  It is more economical to use threads as they share the process resources. Comparatively, it is more expensive and time-consuming to create processes as they require more memory and resources. The overhead for process creation and management is much higher than thread creation and management.

Q32. illustrate the states of thread lifecycle in Java.



Life Cycle of a thread

1. **New Thread:** When a new thread is created, it is in the new state. The thread has not yet started to run when the thread is in this state. When a thread lies in the new state, its code is yet to be run and hasn't started to execute.
2. **Runnable State:** A thread that is ready to run is moved to a runnable state. In this state, a thread might actually be running or it might be ready to 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 and every 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 lie in a runnable state.
3. **Blocked/Waiting state:** When a thread is temporarily inactive, then it's in one of the following states:
   - Blocked
   - Waiting

4. **Timed Waiting:** A thread lies in a 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.
5. **Terminated State:** A thread terminates because of either of the following reasons:
   - Because it exits normally. This happens when the code of the thread has been entirely executed by the program.
   - Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.