JAVA CORE CONCEPTS

JAVA - high-level, class-based, object-oriented programming language.
It is platform independent which means we can write once run anywhere.

JVM- JVM is a engine that provides runtime environment to drive the Java Code or applications.
It converts Java bytecode into machines language. JVM is a part of JRE.
JVM is responsible for allocating memory space

Editor - where we type our program, ex.notepad,eclipse
Compiler - convert program into native machine code
Linker - combining different program files reference in main program.
Loader - loading files from secondary storage like hard disk, flash drive, cd, ram for execution.
Execution - actual execution of code which is handled by OS & processor.

----

public : Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
static : It is a keyword in java which identifies it is class based i.e it can be accessed without creating the instance of a Class.
void : It is the return type of the method. Void defines the method which will not return any value.
main: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
String args[] : It is the parameter passed to the main method.

List any five features of Java?

Some features include Object Oriented, Platform Independent, Robust, Interpreted, Multi-threaded

----

When parseInt() method can be used?

This method is used to get the primitive data type of a certain String.

Why is String class considered immutable?

The String class is immutable, so that once it is created a String object cannot be changed. Since
String is immutable it can safely be shared between many threads ,which is considered very important for multithreaded programming.

What is finalize() method?

It is possible to define a method that will be called just before an object's final destruction by the garbage collector.
This method is called finalize( ), and it can be used to ensure that an object terminates cleanly.

What is an Exception?

An exception is a problem that arises during the execution of a program. Exceptions are caught by
handlers positioned along
the thread's method invocation stack.

Explain Runtime Exceptions?

It is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked
exceptions, runtime exceptions are ignored at the time of compliation.

When throw keyword is used?

An exception can be thrown, either a newly instantiated one or an exception that you just caught,
by using throw keyword.

Define Inheritance?

It is the process where one object acquires the properties of another.
With the use of inheritance the information is made manageable in a hierarchical order.

When super keyword is used?

If the method overrides one of its superclass's methods, overridden method can be invoked through the
use of the keyword super. It can be also used to refer to a hidden field.

What is Polymorphism?

Polymorphism is the ability of an object to take on many forms.
The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

What is Encapsulation?

It is the technique of making the fields in a class private and providing access to the fields via public methods.

If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class.
Therefore encapsulation is also referred to as data hiding.

What is an Interface?

An interface is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Define Packages in Java?

A Package can be defined as a grouping of related types(classes, interfaces, enumerations and annotations)
 providing access protection and name space management.

What do you mean by Multithreaded program?

A multithreaded program contains two or more parts that can run concurrently. Each part of such a program
is called a thread, and each thread defines a separate path of execution.
----

1. OOPS CONCEPTS

  > Object-oriented programming System(OOPs) is a programming pattern based on the concept of “objects” that contain data and methods.
  > The primary purpose of object-oriented programming is to increase the flexibility and maintainability of programs.
  > Object oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works.

OBJECT

  > Object is a bundle of data and its behaviour(we call it as methods).

Example 1:
Object: House
Data: Address, Color, Area
Methods: Open door, close door

Example program:

```
class House {
   String address;
   String color;
   double are;
```

```
  void openDoor() {
    //Write code here
  }
  void closeDoor() {
    //Write code here
  }
 ...
 ...
}
```

CLASS

> A class can be considered as a blueprint using which you can create as many objects as we like.

Example Program:

```
public class Website {

  String webName;
  int webAge;

  Website(String name, int age){     //Constructor
    this.webName = name;             // refers current class webName and webAge
    this.webAge = age;
  }
  public static void main(String args[]){

    Website obj1 = new Website("beginnersbook", 5);
    Website obj2 = new Website("google", 18);

   System.out.println(obj1.webName+" "+obj1.webAge);
   System.out.println(obj2.webName+" "+obj2.webAge);
  }
}
```

Result:

```
beginnersbook 5
google 18
```

CONSTRUCTOR

> Constructor looks like a method but it is not a method.
> It’s name is same as class name and it does not return any value.
> It is used to initialize the object.

> It is invoked at the time of object creation.

```
MyClass obj = new MyClass();
```

> If we look at the right side of this statement, we are calling the default constructor of class MyClass to create a new object.
> We can also have parameters in the constructor, such constructors are known as parameterized constructors.

Example program:

```
public class ConstructorExample {

   int age;
   String name;

   ConstructorExample(){                          //default constructor
          this.name="Chaitanya";
          this.age=30;
   }

   ConstructorExample(String n,int a){     //Parameterized constructor
          this.name=n;
          this.age=a;
   }
   public static void main(String args[]){
          ConstructorExample obj1 = new ConstructorExample();
          ConstructorExample obj2 = new ConstructorExample("Steve", 56);
          System.out.println(obj1.name+" "+obj1.age);
          System.out.println(obj2.name+" "+obj2.age);
   }
}
```

Output:

```
Chaitanya 30
Steve 56
```

OBJECT ORIENTED PROGRAMMING FEATURES

ABSTRACTION

> Abstraction is a process which shows only “relevant” data and “hide” unnecessary details of an object from the user.

Example:

For example, when we login to our bank account online, we enter your user_id and password and press login,
what happens when we press login, how the input data sent to server, how it gets verified is all abstracted away from us.

Another example of abstraction: A car in itself is a well-defined object, which is composed of several other smaller objects
like a gearing system, steering mechanism, engine, which are again have their own subsystems. But for humans car is a one
single object, which can be managed by the help of its subsystems, even if their inner details are unknown.

Example program:

```
abstract class Abs{         //class name must start with abstract.
                            //abstract class have both abstract method and non abstract method
abstract void run();        //In abstract method, there will be no implementation, but we can
pass args like int a,int b.

void print()
{
System.out.println("hello");
}
}

class Newabs extends Abs{

void run()                //we should use all abstract methods in the child class.when we use
abstract method in the child class,
                          we must overwrite the abstract method.
{                                                //non abstract method not compulsion in child
class.
System.out.println("hai im running");
}

public static void main(Strig args[]) {

Newabs obj=new Newabs();
obj.run();
obj.print();
}
}
```

ENCAPSULATION

Encapsulation simply means binding some datas and methods together. If we are creating class, we are doing encapsulation.

Example program: PRIVATE

```
class Privateex {                //Before class we can use only public,abstract,final keywords
                                 //private keyword can be used before variable and method only
private int x=6;                // hiding 'a' from user

public void showDemo()
{
System.out.println("x=" + x);
}

private void testDemo()
{
System.out.println("x=" +x);
}

class AccessEx {
public static void main(String args[])
{
Privateex ad=new Privateex();
ad.showDemo();         // prints x=6
ad.testDemo;           // Cannot print x outside the method
ad.x=6;                //Cannot print x since it is private
}}
```

PROTECTED

```
class ProtectedEx {

protected int x=6;

public void showDemo()
{
System.out.println("x=" + x);
}
protected void testDemo()
{
System.out.println("x=" +x);
}

class Child extends ProtectedEx{
                                         //own properties
}
```

```
class Accessex {
public static void main(String args[])
{
Child ch=new child();
ch.showDemo();        // prints x=6
ch.testDemo;          // prints x=6
ch.x=6;               //prints x=6
}}
```

INHERITANCE

> deriving one class from another class with its own properties
> Can achieve code reusability

SINGLE

```
class parent{
}
class child extends parent{
}
```

Example program:

```
public class parent
{
int sal=10000;
}
class child extends parent          // it can access parent class properties
{
int bonus=5000;
{
public static void main(String args[])
{
child ch=new child();
System.out.println("sal=" +sal);           //prints sal=10000
System.out.println("bonus=" +bonus);    //prints bonus=5000
}
}
```

MULTI LEVEL

```
class parent{
}
class child1 extends parent{
}
class child2 extends child1{
}
class child3 extends child2{          //child3 can access parent,child1,child2 properties with
}                                          with its own properites
```

POLYMORPHISM

poly - many
morphism - behaviour

compline time - method overloading
run time - method overriding

Method overloading : same method name with different arguments inside the same class.

Example program:

```
clsss methodoverload {

void test(){
System.out.println("no parameters");
}
void test(int a){
System.out.println("a= " +a);
}
void test(int a,int b){
System.out.println("a and b = " +a+ "" +b);
}

class overload{
public static void main(String args[]){
methodoverload ob=new methodoverload();
ob.test();
ob.test(5);
ob,.test(2,6);
}
}
```

method overriding : same method name with same arguments in inherited class.

Example program:

```
class Methodoverriding{
public void move(){
System.out.println("animals can move");
}
class Dog extends Methodoverriding{
```

```
public void move(){
System.out.ptintln("Dogs can walk and run");
}
}
class Test{
public static void main(String args[]){
Dog d=new Dog();
d.move();                              //prints latest method
}
}
```

INTERFACE

Only methods and arguments, no implementation

In java, there is no multiple inheritance because of diamond problem occured in C++.
Example:

```
                Ancesstor class
parent1 class                    parent2 class
                   myclass
```

Here, in myclass will get confusion to access ancesstor properties through parent1 class or parent2 class.. thats why they have eliminated multiple inheritance in java and introduced interface
instead of that.

Example program:

```
interface printable{
void print();           //only methods/parameters
void hai();
}

class inter implements printable{
public void print()                      //must overwrite each method from interface and we can also add methods in derived class
{
System.out.println("hello");
}
public void hai()
{
System.out.println("hai");
}

public static void main (String args[]){
inter ob=new inter();
ob.print();
ob.hai();
}
}
```

BASIC CONSTRUCTS IN JAVA

?public: The keyword “public” is an access specifier that declares the main method as unprotected.

static: It says this method belongs to the entire class and NOT a part of any objects of class. The main must always be declared static since
the interpreter users this before any objects are created.

void: The type modifier that states that main does not return any value.

DATA TYPES

boolean either true or false	
char 16 bit Unicode 1.1	- 2 byte
byte 8-bit integer (signed)	- 1 byte
short 16-bit integer (signed)	- 2 byte
int 32-bit integer (signed)	- 4 byte
long 64-bit integer (singed)	- 8 byte
float 32-bit floating point (IEEE754-1985)	- 4 byte
double 64-bit floating point (IEEE754-1985)	- 8 byte

LOOP STATEMENTS

Control Flow Statements in JAVA
while loop
for loop
do-while loop
if-else statement
switch statement

STRING HANDLING

String - Basically string is a sequence of chracters, In java, String is an object which represents sequence of characters.
Java string is used to perform manipulation in the strings.
Java strings are immutable.
In java, String itself a class. we can do operations like comparing the string, finding the length of the string

We can declare string in 2 ways ;

```
String st="value";                              // String literal
String st= new String("newval");             // using new keyword
```

StringBuilder and StringBuffer has created to change from immutable to mutable.

```
StringBuilder st=new StringBuilder("newval");    //Its Mutable, will be fast and not thread safe
StringBuffer st=new StringBuffer("newval");      //Its mutable, little late but thread safe
```

GARBAGE COLLECTION

JVM takes care of garbage collection.

Serial Garbage collector - This garbage collector can work in only one CPU at a time.
It will mark the unreferenced object and delete it then it compacts the live objects.
It uses mark sweep compact algorithm in both younger and older generation garbage collection.

Parallel Garbage collector - This garbage collector can work in multiple machines at a time.
It will uses multiple threads to perform the young generation garbage
collector.
parallel old garbage collection will do in older generation garbage collection.

both garbage collection stops the application.

concurrent mark sweep GC - It will do purely old generation garbage collection.
application will run when garbage collection happens.

JAVA COLLECTION

Collections in java is a framework that provides an architecture to store and manipulate the group of objects.
All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc.
can be performed by Java Collections.
Java Collection simply means a single unit of objects.
Java Collection framework provides many interfaces
(Set, List, Queue, Deque etc.) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

SERIALIZATION

serialization is a mechanism of writing the state of an object into a byte stream.

MULTI-THREADING

threads - threads are light-weight sub process, they share the common memory space.
Multi-threading - The process of executing multiple threads simultaneously is known as multithreading.

The main purpose of multithreading is to provide simultaneous execution of two or more parts of a program to
maximum utilize the CPU time. A multithreaded program contains two or more parts that can run concurrently.
Each such part of a program called thread.

Threads can be created by using two mechanisms :
1. Extending the Thread class
2. Implementing the Runnable Interface