Software Installation:

1. JDK 11
2. Eclipse   
   <https://www.eclipse.org/downloads/packages/release/2023-09/r/eclipse-ide-enterprise-java-and-web-developers>
3. STS  
   <https://www.codejava.net/frameworks/spring/install-spring-tool-suite-windows>
4. Tomcat 9
5. MYSQL / XAMPP [ windows ]

GIT and STS

<https://www.geeksforgeeks.org/how-to-use-git-with-eclipse/>

Practice Questions

<https://www.w3resource.com/java-exercises/>

Advance – competitions [ logical based/ application based, Data structures ]

Arrays, string , collections

LeetCode

HackerRank

hackerearth

PDF or books

Kathy Sierra => OCJP6

Head First => Beginner level

Resources Links

Beginners

Javatpoint

Tutorialspoint

W3schools

Intermediate

Tutorialsteacher

Freecodecamp

digitalocean

Advance

Ofiicial website for java and spring

Baeldung

Medium=> subscription

Linkedin learnings

Udemy

Pluralsight

Coursera

Safari =>

Youtube

Kaushik => java brains

55Minutes

Navin Reddy => telusko

Shalini Mittal => techgatha

Refresher

**Basic Programming Concepts**

1. Java features
   1. Robust
   2. Secure
   3. Platform independent
   4. OOPS
      1. Abstarction
      2. Encapsulation
      3. Polymorphism
      4. Inheritance
   5. Multithreaded
   6. Distributed
2. Compiler and interpreter
   1. Compiler -> checks for syntax errors and if none converts .java to .class[ bytecode or intermediate code understood only by JVM]

Javac => java compiler tool

* 1. Interpreter-> responsible to convert .class file to machine understandable code and execute it
  2. Bytecode verifier
  3. Class Loaders

1. Tokens
   1. Data types
      1. Byte
      2. Char
      3. Short
      4. Int
      5. Long
      6. Float
      7. Double
      8. Boolean
   2. Variables => represent memory locations and the values can vary
   3. Separators
   4. Punctuator
   5. Literals
   6. Keywords
   7. Assignment operator => assignment always happens right to left
2. Variable declaration and initialization
3. Print and println
4. Scanner class to take input from the user
5. Type conversion
6. Pure and mixed expressions
7. Operators
   1. Unary => ++ -- - !
   2. Binary
      1. Arithmetic + - \* / %
      2. Relational < > <= >= == !=
      3. Logical & && | ||
      4. Bitwise & | ^
   3. Ternary ?:
8. Conditional statements
   1. If else
   2. Switch – case
9. Iterative Statements
   1. For
   2. While
   3. Do-while
   4. Break
   5. Continue
   6. Nested loops
10. Arrays
    1. allow to store more than 1 values of similar type
    2. [] syntax
    3. Index starts from 0
    4. new keyowrd to allocate memory

**Object Oriented Programming Concepts**

1. OOPS
   1. What is a class
      1. Blueprint
      2. Template
      3. User defined data type
      4. Composite data type
   2. Create a class
      1. Class syntax
      2. Consists of data members [ attributes/ characteristics/ instance variables] and member methods [ methods or functions that operate on the data and provide with business logic ]
   3. Create object of the class using new keyword
   4. Methods
      1. Can take input => parameters
      2. Provide BL => body of the method within {}
      3. Can return outpur => using return keyword
      4. Signature  
         <access specifier> <return type> <method name>(<0 or more parameters){}  
         public int add(int a, int b){ return a+b;}
   5. Constructors
      1. They are special methods with the name same as the classname
      2. They DO NOT HAVE A return type not even void
      3. Cannot invoke them explicity using '.' operator
      4. Constructors are automatically invoked as soon as the object of the class is created
      5. purpose is to initialize the data members of the class at the time of object creation
      6. which takes no input/parameters => default constructor
      7. if there is no constructor in the class, compiler creates one
   6. constructor overloading / method overloading
      1. More than 1 method with same name but
         1. Different types of parameters
         2. Different number of parameters
         3. Different sequence of parameters
      2. Changing return type is not overloading
2. Access Specifier
   1. private
   2. protected
   3. public
   4. default => is not a keyword for access specifier but if none mentioned its default
3. this keyword
   1. used to resolve naming conflicts if local variable and instance variable name is same
   2. used to call other methods of same class
   3. used to call other constructor of the same class
   4. If using this to call a constructor it has to be the 1st statement in the calling constructor
4. Getters => accessor that allow to access value for a property
5. Setters => modifiers as they modify or update the property
6. toString() => that respresents the java object in the form of string
7. @Override => annotation starts with @ that provides information to the compiler or the JVM
8. Array of objects : Creating array to store user defined entities.   
   Below syntax just creates the array as a container to inform JVM to allocate memory for Student objects  
   Student[] students = new Student[5];  
   To actually store student data, create objects and store at respective locations using index  
   students[0] = new Student(1, “Shalini”);
9. For-each loop => used to iterate but should not use to modify the data
10. Has-A relationship => One class has reference/ dependency on another class.  
    For ex: employee has address  
    Car has engine  
    Book has author  
    Doctor has patients
11. OOPS Principles:
    1. Abstraction : hiding the complexity or implementation details
    2. Encapsulation : binding the data and members and also provide security
    3. Polymorphism : same name different forms
       1. Static/ compile time: overloading  
          Rules for overloading
          1. Can change number of parameters
          2. Can change type of parameters
          3. Can change sequence of parameters
          4. Changing just the return type is NOT OVERLADING
          5. It can be implemented either in same class or parent-child
       2. Dynamic/ runtime : overriding
          1. The signature has to match exactly as that of parent class.
          2. It can only be implemented with parent-child relationship
          3. Even the return type should match except the **Covariant return type**
          4. Cannot restrict the access level in the child class
          5. Cannot throw an exception that is less specific to the parent class
    4. Inheritance : Reusability
12. String : explored various method. VVIMP use equals to compare 2 strings
13. Static modifier :
    1. Also called as class variables
    2. static is a modifier => instance variables, methods, inner classes
    3. CANNOT use on local variables or outer class
    4. static members are also called as class variables => global data
    5. static members do not require object of the class rather can directly be accessed using classname
    6. static methods can directly access other static members but can access non-static members only via objects
    7. non-static members can access other static or non-static members directly
    8. this keyword is not accessible inside a static method

**Inheritance**

1. use the extends keyword
2. Whenever object of the child class is created, first the topmost parent object is created and flows down the hierarchy. The constructor of topmost parent is called first
3. Child class inherits all the members of the class except private members and constructor
4. Super keyword is used
   1. To call the parent class constructor
   2. To invoke the parent class members
5. If parent class has no constructors at all then child class can be created without explicitly making a call to super()
6. If parent class has a parameterized constructor and no default constructor then child class has to mandatorily call the parameterized constructor using super passing in the values for the parameters of the parent constructor
7. Hence we say do create a default may be an empty constructor if parent class has even one parameterized constructor
8. Overriding
9. Object class => is the super class which all class by default inherit irrespective we write extends Object or not
10. Memory : Stack , heap and string pool
11. Parent class reference can store object of its class os subtype  
    class A{}  
    class B extends A{}  
    class C extends A{}  
    A a ; // a is the parent class reference  
    a = new A();// a can point to object of its own class  
    a = new B();// a can point to object of its child class  
    Reference decides which methods can be called  
    Object decides , method of which class will be invoked
12. Dynamic Polymorphism and suggested to not downcast from super class as it may end up with ClassCastException
13. Abstract classes
    1. abstract is a modifier
    2. Can be used on class and methods
    3. If a class is declared as abstract :
    4. you cannot create the object of the class using new/ instantiate
    5. abstract classes may or may not have abstract methods but if a class has even one abstract method then the class needs to be declared as abstract
    6. abstract methods do not have implementations
    7. abstract methods needs to be implemented by the respective child classes or child classes should declare them as abstract
14. Final Modifier
    1. Variable declared as final cannot be modified, used to create variables to store constant values
    2. Methods declared as final cannot be overridden
    3. Classes declared as final cannot be inherited
15. Interfaces
    1. They are like contracts
    2. By default abstract
    3. Can have only public static final variables
    4. Can have public abstract methods
    5. Since 1.8 , can have default and static methods with keywords default and static respectively
    6. Default methods can be optionally overridden by the implementation class
    7. Classes always implement the interfaces and provide the implementation of the abstract methods or declare themselves as abstract
    8. Interfaces can extend other interfaces
    9. Cannot create object of interface
    10. Reference of type interface can point to object of its implemention class  
        interface Payment{void pay();}  
        class CreditCard implements Payment{ void pay(){}}

class NeftCard implements Payment{ void pay(){}}  
Payment p = new CreditCard();

p = new NeftCard();

**Collections  
  
**

1. Generics
   1. Use <> [ diamond brackets ] where you can specify the type of items collection stores  
      List<Integer> list = new ArrayList<>();
   2. For-each loop with collections