

Session Timings
CST sun-thu | 6.30pm-7.30pm
IST mon-fri | 5am-6am

Setup

Download Java 17
https://learn.microsoft.com/en-us/java/openjdk/download

Download STS IDE
https://spring.io/tools

---

what is java?
- a high level language (one of machine language, assembly language, and high-level language)
- object oriented language
  - eveything in java is object which has a state and behaviour
- statically typed language
- garbage collected language
- mutithreaded language
- both compiled and interpreted
  - .java(source code) --compile|using-javac-> .class(byte code), --interpret|using-jvm-> machine code
- java programs are portable platform independent
  - compile once and execute anywhere using jvm
  - .java and .class files are machine independent and therefore portable
- secure language
  - no explicit use of pointers (no direct access to memory like c)
    - helps resolve probelms like memory leakage when garbage is not cleared
  - jvm verfies byte-code eg. type safety, inheritance, naming rules etc

JDK, JRE, and JVM
Java Development Kit - JDK
- JDK = JRE + compiler(javac)

Java Runtime Environment - JRE
- JVM + some libraries and other files used by JVM to run byte code

Java Virtual Machine - JVM
- loads, verifies, and exectures byte code (.class files) is executed
- any language code that is converted to bytecode can be run with it

---

SOME JAVA COMMANDS

```
        java -version //to check cuurently used java version

        javac JavaFileName.java // to compile
        java JavaFileName // to run
```

--learn more in java-indepth section

THE MAIN METHOD

```
        the entry point for JVM to start execution
        main method can accept arguments
                either seperated by spaces
                or arguments that have space in them can be enclosed in double quotes
```

--learn more in java-indepth section

COMMENTS

```
        single line //
        multi line /* */
        documentation /** */
```

NAMING CONVENTION

```
        case sensitive
        class and interface names
                start with upper case letter
                eg. SampleClassForCommandLineProject
        method names
                start with lower case letter
                generally a verb
        variable names
                start with lower case letter
                rules to declare variable names
                        alphabets, numbers, _ and $
                        first char cannot be number
                        not reserved keywords
                        no blank spaces
                        no limit for length but keep it short and readable
        contants
                all upper case with underscore for readability
        program file name should match the "public" class name
        package name
                all lower case
                eg. com.amazon, org.mit.edu, com.google
```

java follows CamelCase naming convention

JAVA KEYWORDS

```
        set of reserved words by the programming labguage itself
        these cannot be used for user defined identifiers
        https://www.w3schools.com/java/java_ref_keywords.asp


------------------------------

DATA TYPES

Primitive
        Data Type       Default Value   Default size
        boolean         false           1 bit   (stores true or false values)
        char            '\u0000'        2 byte  (stores single charcter/letter ascii or utf-8 values)
        byte            0               1 byte  (stores whole numbers from -128 to 127)
        short           0               2 byte  (stores whole numbers from -32,768 to 32,767)
        int             0               4 byte  (stores whole numbers from -2,147,483,648 to 2,147,483,647)
        long            0L              8 byte  (stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
        float           0.0f            4 byte  (stores fractional numbers. Sufficient for storing 6 to 7 decimal digits)
        double          0.0d            8 byte  (stores fractional numbers. Sufficient for storing 15 decimal digits)


Non Primitive
        String, Array, Classes, Enum etc..

Some References
ASCII - American Standard Code for Information Interchange (7-bit size)
UTF-8 - Unicode Transformation Format 8 - (1 to 4 Bytes size)
        https://www.fileformat.info/info/charset/UTF-8/list.htm

------------------------------

VARIABLES

        name given to a reserved memory location
        naming, declaration, initialization, accessing

        data_type var_name;
        var_name = value;
        data_type var_name = value;


------------------------------

OPERATORS

        Type            Category                        Precedence(top to bottom)
        Unary           postfix                         expr++ expr--
                        prefix                          ++expr --expr +expr -expr ~ !
        Arithmetic      multiplicative                  * / %
                        additive                        + -
        Shift           shift                           << >> >>>
        Relational      comparison                      < > <= >= instanceof
                        equality                        == !=
        Bitwise         bitwise AND                     &
                        bitwise exclusive OR            ^
                        bitwise inclusive OR            |
        Logical         logical AND                     &&
                        logical OR                      ||
        Ternary         ternary                         ? :
        Assignment      assignment                      = += -= *= /= %= &= ^= |= <<= >>= >>>=
```

-precedence in an expression goes from right to left.

Practice from here
https://www.scientecheasy.com/2021/04/operators-interview-questions.html/
https://www.geeksforgeeks.org/java-operators-question-1/

TYPE CASTNG

2 types
	Widening | Automatic | smaller to larger type size
		byte -> short -> char -> int -> long -> float -> double

	Narrowing | Manual | larger to smaller type size
		double -> float -> long -> int -> char -> short -> byte

```
long l =  9,223,372,036,854,775,807l;
int i = (int) l;
```

Array

collection of elements of same data type
index based starting from 0 to n-1 indexes for n elements
fixed in size

ways to declare, instantiate(create), initialize 1-D array in java
	data_type[] arr; //declare only
	data_type []arr; //declare only
	data_type arr[]; //declare only
	data_type[] arr = new data_type[size] //declare and instantiate
	data_type[] arr = new data_type[size]{val1, val2, val3....} //declare, instantiate, initialize
	data_type[] arr = {val1, val2, val3....} //declare, instantiate, initialize
	arr[index] = value //initialization | assigning value to an array element

accessing (usage)
	direct access via index print(arr[index])
	traverse using a loop eg. for loop

passsing array as method argument
	sum(int[] arr)

2-D array
	data_type[][] arr = new data_type[x][y];
	data_type[] arr = { {val1, val2, val3....}, {val1, val2..} }

CONTROL STATEMENTS

decision making
	if-else
	switch-case
loop
	for
	while
	do-while
	for-each
jump
	continue
	break

PROGRAM INPUT AND OUTPUT

output
	System class
input
	Scanner class

ASSIGNMENT-1

write a program to print command line argumets with words containing spaces
wap to print minimum and maximum values of all the primitve data types
wap to demonstrate the usage all the operatars in java
wap to demostrate widening and narrowing for all possible primtive data types
wap to print even and odd numbers between a given range a, b
wap to print multiplication table of a given number
wap to find the greatest number in a given array
wap to find grade for given marks
write functions to find sum, diff, multi, div of two given numbers
wap to print duplicate elements in a given arr
wap to print two number with even occurences in a given array
wap to print sum of even and sum odd numbers of first N numbers where N is input
wap to sort an array in ascending and descending order
wap a program to search an element and in array
wap to round off a number to given number of decimal plcaes using String and DecimalFormat class
wap to reverse a string
wap to print a given string in lowercase

Note that you should not hard code input and rather accept program inputs from console using Scanner class and do not hard code them in your program

---

OBJECT ORIENTED PROGRAMMING

mimicking the environment around us using code

a computer programming model that organizes software design around data, or objects, rather than functions and logic
with the help of concepts like classes, objects, state, behaviour, inheritance, data hiding, abstraction, polymorphism, message passing...

METHOD

a collection a statements meant to perform specific task
method signature = method name + parameter list
used by jvm to identify methods

VARIABLE

name of the memory location used for storing and accessing data

---

CLASS

user-defined data type
a blueprint of an object (real life entities)
consists of data memebers and member functions

OBJECT
basic unit in OOP
represents real life entities
instance of a class
has 3 main properties
identity
name of the object
state
its attributes
using variables (data members or instance variables)
behaviour
actions
using methods (member functions or instance methods)

STRUCTURE OF A TYPICAL JAVA CLASS

class name
fields
methods
constructors
default and/or parametrized

---

new KEYWORD

create an instance of the class i.e a new object
allocates the memory at runtime
returns referece to that memory

```
        invokes object constructor


-------------------------


STEPS TO CONSTRUCT AN OBJECT

        0. gather info on an object, take vehicle for example
        1. write a class with properties and functions available for that object
        2. declare and define an object using new keyword and class constructor method

EXAMPLE: Vehicle
        a vehicle has state information like, number of name, price, wheels
        a vehicle has behaiours like run and stop


-------------------------

this KEYWORD

        refers to the current class instance (object)
        can be used to access crrent class instance variables implicitly
        can be used to invoke crrent class instance methods implicitly

-------------------------


VARIABLE TYPES

        local
                declared and defined inside a method
                destroyed after method execution
                cannot be accessed from outside the method block
        instance
                declared at class level
                are accessible to all the methods inside the class
                will have one copy for each object instance
        static
                declared at class level
                are accessible to all the methods inside the class
                will have same copy for all object instances
                can be access without even created objects instances

class
        var1
        var2
        static var3

object1
        var1@addr11
        var2@addr12
        var3@addr3
object2
        var1@addr21
        var2@addr22
        var3@addr3


-------------------------
```

PACKAGES

	group of classes, sub packages and interfaces
	prevents naming conflicts between duplicate class names
	package level access control
	clear code organisation

	package names are in lowecase and sub packages are seperated by "."

	import package.name.Class; // to import a single class
	import package.name.*; // to import the whole package

eg.
	import java.util.Scanner;
	import java.util.*;

com.tutorials
com.tutorials.pojo or com.tutorials.model or com.tutorials.entity
com.tutorials.pojo or com.tutorials.model or com.tutorials.entity
com.tutorials.service
com.tutorials.dao
.....

ACCESS MODIFIERS
	helps restrict scope of a class, constructor, variable, method, or data member

4 types

	default/no-mod	private	protected	public
same class	Yes	Yes	Yes	Yes
same package sub class	Yes	No	Yes	Yes
same package non sub class	Yes	No	Yes	Yes
different package sub class	No	No	Yes	Yes
different package non sub class	No	No	No	Yes

classes
	public or default
variables, methods and constructors
	default/no-mod private protected public

NON ACCESS MODIFIERS

	provide information about the characteristics of a class, method, or variable to the JVM

7 types
	static - variables and methods
		 variables that will exist independently of any instances created for the class
		 methods that will exist independently of any instances created for the class

	final - variables, methods and classes
		 finalizing the implementations of classes, methods, and variables

```
                final variable can be explicitly initialized only once
                final method cannot be overridden by any subclasses
                final class cannot be extended

        abstract - classes and methods
                abstract class can never be instantiated and can only be extended
                a class cannot be both abstract and final (since a final class cannot be extended)
                abstract class may contain both abstract methods as well normal methods
                method declared without any implementation is abstarct method
                any class that extends an abstract class must implement all the abstract methods of the super class, unless the subclass is also an abstract class
                if a class contains one or more abstract methods, then the class must be declared abstract
                an abstract class does not need to contain abstract methods

        synchronized - method
                a synchronized method can be accessed by only one thread at a time

        transient - variable
                skip it when serializing

        volatile - variable
                chnages on one thread are reflected across all threads
                value of a volatile variable is always read from the main memory and not from the local thread cache, and it helps to improve thread performance

        native - method
                indicates that the method is implemented in a language other than java
                and can be executed from java using JNI



-------------------------

GETTERS AND SETTERS
        will learn with encapsulation

-------------------------


OOP PRINCIPLES

INHERITANCE

        mechanism by which one class is allowed to inherit the features (fields and methods) of another class
        using extends keyword
        is-a relationship
        super and sub class
        reusability

        types: single 1-2, multi-level 1-2-3, hierarchical 1-2.1,-2.2, multiple 1,2=3(not allowed in java), hybrid/circular dependencies(not allowed in java)

        vehicle, car, grandfather, father, son

        class A msg(), class B msg(), class C extends A,B, new C().msg(); //Now which msg() method would be invoked?
        ambiguity and circular dependencies incase of hybrid inheritance


POLYMORPHISM

        having many forms
        two types:
                method-overloading/compile-time
                        change in number of and type of method arguments
```

```
                eg. multiply int, double
        method-overriding/runtime
                call to the overridden method is resolved at runtime



ENCAPSULATION

        data hiding
        bundling of data and methods - class
        way of hiding the implementation details of a class from outside access
        and only exposing a public interface that can be used to interact with the object

        declare instance variables of a class as private
        define public methods called getters and setters

        eg. BankAccount with private balance field and credit, debit and getBalance public methods

        read only, write only classes




ABSTRACTION

        implementation hiding
        process of identifying only the required characteristics of an object ignoring the irrelevant details
        eg. product of two number -> a.b, sum a for b times

        used for writing specifications
        in java using abstract class, abstract method and interfaces

        abstract class:
                declared with abstract keyword - abstract class ClassName
                can have both abstract and concrete methods
                can have variables and constants
                sub class should extend abstract class
                sub class should override abstract methods
                can have constructors but cannot create objects of an abstract class
                eg. calculator with add, sub, multiple, divide

        interface: 100% abstraction
                declared with interface keyword - interface InterfaceName
                says what a class must do and not how
                can have only abstract methods
                can have only constants
                all the methods are public and abstract, and all the fields are public, static, and final by default
                a class should implement an interface
                the implementing class should override all the abstract methods
                abstract keyword is not necessary for methods and are abstract by default



------------------------


OBJECT class

        super class of all the classes by default in java
        methods:
                getClass(), hashCode(), toString()...
```

ASSIGNMENT-2

explain what a class with an example in java
explain what an object/instance is with an example in java
wap to differentiate between local, instance and static variables in java
what is the purpose of new keyword in java?
what are access modifiers in java?
wap to demonstrate usage of default, public and private access modifiers ion java
what are non access modifiers in java?
wap to demonstrate usage of static and final non access modifiers in java
what are the 4 major principles in object oriented programming?
what is inheritance in java? when/why is it used?
wap to demonstrate single, multi-level, and hierarchical inheritance in java
wap to demonstrate usage of this and super keywords in java
does java support multiple and hybrid inheritance? justify your answer
what is polymorphism? what are its types in java? when/why is it used?
explain why method-overloading is aka compile-time polymorphism in java
wap to demonstrate method-overloading in java
explain why method-overriding is aka runtime-time polymorphism in java
wap to demonstrate method-overriding in java

CALL BY VALUE AND CALL BY REFERENCE

- in call by value, the modification done to the parameter passed does not reflect in the caller's scope
- in the call by reference, the modification done to the parameter passed are persistent and changes are reflected in the caller's scope

- java is passed by value
  - primitives are passed as value
  - for objects its reference is passed as value

Some built-in java classes

- Math
  - sqrt(), pow(), min(), max(), avg(), sin(), cos()...
- Wrapper Classes
  - Integer, Double, Boolean ...
  - convert primitive into object and object to primitive
  - autoboxing and unboxing
  - immutable - creates a new object upon value change
- Date
  - LocalDate, LocalTime, LocalDateTime.. now()..
- String
  - is an object that represents sequence of charaters. or simply an array of char

```
		create using double quotes or using new keyword
			only one literal/string-instance to string constant pool
		length(), equals(), split(), replace(), substring(), isEmpty(), isBlank(), indexOf(), toLowerCase(), toUpperCase(), trim()...
		conactenation - +, conact(), String.format(), String.join()
		immutable
		implements Serializable, Comparable and CharSequence interfaces



------------------------

EXCEPTION HANDLING

	handle the runtime errors such as NullPointerException, ArrayIndexOutOfBoundsException, SQLException etc...
	java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception (RuntimeException--NullPointer..) and
Error(StackOverflowError)

	Checked Exceptions
		classes that directly inherit the Throwable class except RuntimeException and Error
	UnChecked Exceptions
		classes that inherit the RuntimeException
		NullPointerException, ArrayIndexOutOfBoundsException...

	Error | irrecoverable
		OutOfMemoryError, VirtualMachineError, AssertionError etc


	important clsases to remember: Throwable, Exception, Error, RuntimeException

How To?
	using try, catch, finally, throw, throws, try-catch, try-catch-catch, try-finally, try-catch-finally
	custom exception by extending Exception class


Eg.
int data=100/0;  //ArithmeticException
String s=null; System.out.println(s.length());//NullPointerException


------------------------

COLLECTIONS FRAMEWORK
	storing and manipulating a group of objects in a data structure
	ARRAY LIST
	HASH MAP
	HASH SET


------------------------

STREAM API
	used for various operations especially on collections
	filter, sort, etc

	properties
		stream does not store elements
		stream is functional in nature. ops on a stream does not modify it's source
		elements of a stream are only visited once during the life of a stream

	intermediate (map, filter...) and terminal(collect, forEach...) operations
```

```
eg.
list.stream().filter(a -> a.pages > 0).map(a->a.pages).collect(Collectors.toList())
list.stream().filter(a -> a.pages > 0).forEach()..
        Collectors.summingInt(a->a.pages)
        Collectors.toSet()
        Collectors.toMap(a->a.title, a->a.pages)
        count()

------------------------


GENERICS

        to allow type to be a parameter to methods, classes, and interfaces
        why not Object class? - type safety eg. setting any type values to instance fields
        generic class and methods
        naming convention for type params
                T - type, E - element, K - key, V - value
        example object definition with generics
                BaseType <Type> obj = new BaseType <Type>()
        generally used to create type safe data structures


------------------------


EXTERNAL LIBRARIES

        libraries - collection of classes that have been written by somebody and can be imported in our project for use
        the .jar file | java archive

        eg. Google Guava for MultiMap
        Steps:
                download the library jar file from mvnrepository.com
                right click on your project inside your IDE
                Build Path -> Configure uild Path -> select Classpath -> click add external JARs -> slect your .jar file -> apply and close
                now you can start using library classes and methods


------------------------


MAVEN
        dependency and build management tool
        comes pre installed with popular IDEs like Eclipse, STS
        current version is 3.9...
        uses pom.xml for configuration of dependency or build instructions
        helps us in automating import of external libraries, building and deploying projects

        local(.m2), central(https://repo.maven.apache.org/maven2/) and remote(in-house) repository
        search for maven repos https://mvnrepository.com/ or https://central.sonatype.com/?smo=true

        Dependency Management
        steps:
                create a maven project from your IDE
                        IDE adds pom.xml and created the right maven project structure for you
                search for your external library on on google with maven as prefix or on https://mvnrepository.com/
                copy pom configuration and add it to depdendency section of pom.xml
                now right click on project -> Maven -> Update -> Force Update
                done! start using your libraries classes and methods
```

Build Management

Pre-requisite - Basic SQL

JAVA DATABASE CONNECTIVITY - JDBC

- a java library to connect and query to databases
  - a database is used to store application/software/user data
- there 2 types of databases Relational and Non Relational or SQL and NoSQL
- eg. MySQL, PostgreSQL, MongoDB, Redis etc

- JDBC API
  - provides methods and interfaces to connect with databases through JDBC driver
- JDBC drivers
  - java needs jdbc drivers to connect to these databases
  - generally every database have their own JDBC librries

Interfaces of JDBC API
- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

Classes of JDBC API
- DriverManager class
- Blob class
- Clob class
- Types class

ORM - JPA and HIBERNATE

- Object Relational Mapping - ORM
  - is a layer that connects object oriented programming to relational databases
- Java Persistence API (now Jakarta Persistence API)
  - is a specification to facilitate object-relational mapping to manage relational data in Java applications
  - you can work directly with objects instead of using raw sql queries
- Hibernate
  - is a java ORM framework
  - it implements JPA

- JPA Concepts
  - Entity
  - Mappings

```
                JPQL
                Criteria API
        Hibernate Concepts
                +above JPA concepts
                Architecture Elements
                        SessionFactory
                                used to get session object
                        Session
                                a wrapper around JDBC
                                provides interface to connect between application and database
                                provides methods to insert, update and delete objects
                                provides some factory methods such as to create transaction etc..
                        Transaction
                                a unit of work or operations performed on a database
                        ConnectionProvider
                                to get connection object
                        TransactionFactory
                                to get transaction object
                Configuration
                Dialects
                Transaction management
                HQL and HCQL
                Cache

        Maven dependecies
                hibernate-core
                mysql-connector-java

sample code ref: https://www.mastertheboss.com/hibernate-jpa/maven-hibernate-jpa/maven-and-hibernate-4-tutorial/

------------------------

DAO Pattern

        Data Access Object pattern
        is a strtuctural pattern that allowes application layer to be isolated  from persisternce layer using abstraction


------------------------

MUTI-THREADING

        Process
                an instance of a program execution
                is nothing but an executing program
        Thread
                a light-weight process
                basic unit of execution within a given process
                one process can have many threads

        Java is single process and single threaded by default

        Mutithreading?
                is to seperate time consuming tasks eg. sorting or calcuating even and odd numbers from billions of numbers
                or seperate tasks that might interfere execution of other tasks eg. file download, sending an email
                it is the ability of the cpu to exceute multiple tasks concurrently using threads

                advantages
                        better performance
                        better resource utilization
                        better user exeperience with responsive apps
```

```
        disadvantages
                syncronization is little complex
                not easy to test (asyn codes)
                it is expensive to create new threads all the time
                number of threads a cpu can handle is limited and by memory and switching time
                after a saturation point increasing threads will decrese performance

        rule of thumb : number of threads <= cpu cores for optimum results
        AND do not use unless required

        can a single core CPU have multiple threads?
                yes of course! hundreds of threads are not a problem
                just go into your windows task manager.

        how can a single cpu core handle multiple threads?
                using time slicing algorithm
        who decides what thread to be executed?
                thread scheduler generally using preemptive scheduling
                first cme first serve and based on thread priority

PARALLEL PROGRAMMING
        multiple cores executing multiple threads
        single core can not help you much in terms in performance by using multithreading


Thread Lifecycle
        new state
                thread is in this state util we call strt() method
        active state
                when we call start()
        blocked/waiting state
                when a thread is waiting for another trhead to finish
        terminated state
                when a thread has finished its task

Threads can be created in 3 ways in java
        extending thread class
                not recommended because you cannot extend any other classes
        implementing runnable interface
        using anonymous class


STACK MEMORY
        local variables
        method arguments
        method calls
        each thread has its own stack memory

HEAP MEMORY
        objects
        static variables
        live as long as it is referenced from somewhere in the application


good read: https://jcip.net


------------------------
```

ASSIGNMENT-3

write crud operations on film table with FilmHibernateDAO
complete the implementation of ActorJdbcDAO
write FilmJdbcDAO
write crud operations on actor table with ActorJdbcDAO
write crud operations on film table with FilmJdbcDAO

create a new table in database called employee with columns id, first_name, last_name, age, role
write Employee entity, confgure it in hibernate.cfg.xml, write EmloyeeHibernateDAO
write crud operations in java using EmloyeeHibernateDAO

THE WEB

- the internet
  - multiple computers communicating each other
- http - hypertext transfer protocol
  - a set of rules that defines how to transfer data between devices over the network
  - is a client-server protocol
- servers
  - a software that uses http or other protocols to respond client requests
  - tomcat, netty, jetty..
- client
  - a software that uses http or other protocols to request data from server
  - web browsers, mobile apps..

- front-end
  - user interface or how a web page looks
  - html, css, js, react...
- back-end
  - server side software
  - java, python, php, golang...
- database
  - where user data is stored
  - MySQL, Postgres, Mongo....

HTTP

- main features
  - connectionless - no prior checks to see if the other device is available
  - media independent - text, documents, audio, video etc
  - stateless - two devices know each other only for the current request and response cycle
- a http transaction involves request and response messages

- Uniform Resource Locator - URL
  - a standard way of specifying or accessing a resource on the internet
  - syntax: <protocol>://<host><:optional-port>/<path>
  - eg. https://www.google.com/search

- Request
  - sent by the client to servers

```
        parts of HTTP Request
                request line, set of headers, a blank line, optional body
        syntax. <http-method>/<url-path> <http-version>
                <headers..>
                <headers..>

                <body>

        eg1.    GET /search HTTP/1.1
                Host: google.com
                Content-Type: text/html

        eg2.    POST /search HTTP/1.1
                Host: google.com
                Content-Type: application/json

                {
                        "bname": "abcd",
                }

Response
        sent by the server back to client
        parts of HTTP response
                status line, headers, a blank line, optional body
        syntax. <http-version> <status-code> <status-message>
                <headers..>
                <headers..>

                <body>
        eg.     HTTP/1.1 200 OK
                Content-Type: text/html

                Hello World

HTTP Methods
        GET - to retrieve data
        HEAD - identical to GET but without response body
        POST - to submit data
        PUT - to update data fully
        PATCH - to partially update data
        DELETE - to delete data
        OPTIONS - to get avialable http methods on an url

HTTP Status Codes
        Informational responses (100 – 199)
        Successful responses (200 – 299)
        Redirection messages (300 – 399)
        Client error responses (400 – 499)
        Server error responses (500 – 599)

        200 OK - success upon get, head, put, post
        201 Created upon post or put
        400 Bad Request
        401 Unauthorized
        403 Forbidden
        404 Not Found
        405 Method Not Allowed
        500 Internal Server Error
        503 Service Unvavailable

HTTP Headers
```

```
                used by client and server to exchange additional information
                case insensitive

                Content-Type: text/html
                Cookie: name=value; name2=value2; name3=value3



---------------------------


WEB DEVELOPMENT STACK

        group of languages or frameworks or libraries or tools that developer or group of developers use to build a web application

        some examples are:
                html, css, js, java, mysql
                react, java, mysql
                jsp, java servlets, mysql
                html, css,  java servlets, mysql
                html, css, js, java spring boot, mysql

---------------------------


SERVLET

        a technology/library/interface/class etc in java used to write web applications
        servlets are executed inside web servers and respond to http requests
        web servers for example tomcat, netty, jetty, jboss etc

        Servlet API
                provides two packages javax.servlet and javax.servlet.http
                interfaces and classes javax.servlet
                        Servlet, ServletRequest, ServletResponse, RequestDispatcher, ServletContext ....
                        GenericServlet, ServletInputStream, ServletOutputStream, ServletException ....
                interfaces and classes javax.servlet.http
                        HttpServletRequest, HttpServletResponse, HttpSession ...
                        HttpServlet, Cookie, HttpServletRequestWrapper, HttpServletResponseWrapper ...


        HttpServlet class
                provides methods to work with http protocol
                like doGet, doPost, doPut ...

        Servlet Life Cycle
                Servlet class is loaded (only once)
                        when first request is recieved by web container(tomcat)
                Servlet instance is created (only once)
                        web container creates it
                init method is invoked (only once)
                        web container calls init method containing code for db set up etc..
                service method is invoked
                        web container calls the service method each time when request is received
                destroy method is invoked
                        destroys the servlet instance


        3 ways to code servlets
                by implementing servlet interface
                by inheriting GenericServlet class
```

```
                by inheriting HttpServlet class (mostly used)


        directory structure of servlet project
                project-name(folder)
                        WEB-INF(folder)
                                classes(folder)
                                        .class(files)
                                web.xml(file)
                                lib(folder)
                        html, css, js, images...(files)

        deployment descriptor(web.xml)
                an xml file used by web container to get information about servlets


        project: rest-api crud with java servlets, hibernate and mysql
        porject: integrating rest-api crud with front end


--------------------------------------------------------------------------------




SERVLET

        a technology/library/interface/class etc in java used to write web applications
        servlets are executed inside web servers and respond to http requests
        web servers for example tomcat netty, jetty etc

        Servlet API
                provides two packages javax.servlet and javax.servlet.http
                interfaces and classes javax.servlet
                        Servlet, ServletRequest, ServletResponse, RequestDispatcher, ServletContext ....
                        GenericServlet, ServletInputStream, ServletOutputStream, ServletException ....
                interfaces and classes javax.servlet.http
                        HttpServletRequest, HttpServletResponse, HttpSession ...
                        HttpServlet, Cookie, HttpServletRequestWrapper, HttpServletResponseWrapper ...


        HttpServlet class
                provides methods to work with http protocol
                like doGet, doPost, doPut ...

        Servlet Life Cycle
                Servlet class is loaded (only once)
                        when first request is recieved by web container(tomcat)
                Servlet instance is created (only once)
                        web container creates it
                init method is invoked (only once)
                        web container calls init method containing code for db set up etc..
                service method is invoked
                        web container calls the service method each time when request is received
                destroy method is invoked
                        destroys the servlet instance


        3 ways to code servlets
                by implementing servlet interface
```

```
                by inheriting GenericServlet class
                by inheriting HttpServlet class (mostly used)


        directory structure of servlet project
                project-name(folder)
                        WEB-INF(folder)
                                classes(folder)
                                        .class(files)
                                web.xml(file)
                                lib(folder)
                        html, css, js, images...(files)

        deployment descriptor(web.xml)
                an xml file used by web container to get information about servlets


---------------------------


REST APIs

        /actor/{id}
                GET, POST, PUT, DELETE methods

        4 Richardson maturity levels
                level 0 - evrything using post request
                level 1 - get and post
                level 2 - all https methods/verbs to distinguish between actions
                level 3 - you follow HATEOS


---------------------------


HTTP NAMING CONVENtIONS

        url - small case, seprated by - incase of mulitple words, seperated by / incases of paths
        url parameters - follow the same as url


---------------------------

PROJECT 1 - rest-api crud with java servlets, hibernate and mysql

---------------------------

PROJECT 2 - integrating rest-api crud with front end

--------------------------------------------------------------------------------


---------------------------

Some background...


Inversion of Control - IoC
        transfer of the control of objects or portions of a program to a container or framework
        can be achieved using patterns like factory, dependency injection etc
```

Dependency Injection
	connecting objects with other objects, or “injecting” objects into other objects
	loose coupling
	thus two types of dependency injection in java
		Constructor Injection
		Setter Injection

SPRING FRAMEWORK

	framework of frameworks
	provides support to various frameworks such as Hibernate etc
	easy to test, fast development, powerful abstraction
	built as a IoC framework

Spring Modules
	Data Access
		JDBC, ORM, JMS, Transactions
	Web
		web, servlet, struts, portlet
	AOP
	Aspect
	Instrumentation
	Spring Core Container
		core and beans - IoC and Dependency Injection features
		context - I18N, EJB, JMS
		expression language
	Test

Spring IoC Container
	ApplicationContext interface represents the IoC container in spring
	responsible for instantiating, configuring and assembling objects known as beans
	also manages bean life cycles

DI in Spring
	can be done through constructors, setters or fields
	using @Autowired annotation

	Autowire Field
	class Employee{
		@Autowired
		Address address;
	}

	Autowire Setter
	class Employee{
		Address address;

		@Autowired
		void setAddress(Address address){
			this.address=address;

```
        }
    }

    Autowire Construction
    class Employee{
        Address address;

        @Autowired
        Employee(Address address){
            this.address=address;
        }
    }


    the address bean needs to be configured using xml file, @Bean or @Component annotations



---------------------------

SPRING BOOT | SPRING WEB and SPRING JPA

    is a project that is built on the top of the Spring Framework
    rapid application development
    embedded server helps with creating a single standalone deployable jar
    no more boilerplate xml configuration
    production-ready features such as metrics, health checks, and externalised configuration
    it provides starter libraries with some production default configuration for various spring modules


---------------------------

PROJECT 3 - rest-api crud with spring boot, spring jpa and mysql

---------------------------

PROJECT 4 - TODO LIST using java spring boot, html, css and js


--------------------------------------------------------------------------------







JVM

    jvm is a specification
    different orgs or companies like oracle, microsoft, redhat etc have their own implementation jvm.
    these companies provide both free and commericial jvm for use

    its job is to load, verify, and execute byte-code
    architecture: major components: classloader, memory area, execution engine, jni
    1. class loader
        loads the .class files (the byte code) into jvm when required
        verifies the code eg. type checking etc
        initialized any static vars or execute static blocks
    2. memory area
        2.1 class/method area
```

```
                        all class level data
                        static variables
                        only one per jvm
                        not thread safe
                2.2 heap area
                        objects, instance variables, arrays
                        one per jvm
                        not thread safe
                2.3 stack area
                        stack frames
                        one stack frame per thread
                        one stack frame per method lifecycle
                        thread safe
                2.4 pc registers
                        one per thread
                        address of current executing instructions
                2.5 native method stack
                        per thread
                        native method information
        3. execution engine
                3.1 interpreter
                        reads bytecode and executes piece by piece
                        even if the same method is executed twice, it will be interpreted twice
                3.2 JIT compiler
                        finds repeated code keeps it converted to native code to save it from interpretation effort
                3.3 garbage collector
                        removes unreferenced objects
                        can be explicitly invoked using System.gc() but not gauranteed because of thread priority

        4. JNI
                helps interact with native method libraries



-------------------------------------


STATIC BLOCKS

        executed when the class is loaded in memory


so can you run a java class without main method??
        yes but no
        it was possible till java 5



-------------------------------------

ANONYMOUS CLASSES

        inner classes with no name
        since no name, cant create instance or objects
        are declared and used on the spot
        can be created either extending a class or implementing an interface
        syntax:
                new ClassNam(){
                        @Override
```

```
                        method_impl...
                }
                new InterfaceName(){
                        method_impl...
                }
```

ASSOCIATION

        another important OOP concept
        two types
        1. IS-A
                car IS A vehicle
                bike IS A vehicle
        2. HAS-A
                employee HAS An address
                student HAS An address

SYNCRONIZATION

        in a multithreading environment, synchronization is tha capability to control the access of multiple threads to any shared resource to prevent thread interference or a consistency problem

        the problem: objects in java are stored in heap area and are therfore not thread safe

        the concept:
        mutual exclusion principle (no two concurrent events)
        every object has a lock associated with it
        a thread that needs 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.
        and one of the way to achieve this is by <synchronized> keyword in java

        solution: we can use <synchronized> keyword on a method to lock for its use

        examples of without and with synchronization

VOLATILE

        problem: jvm or cpu may cache variable values for faster access and this inturn makes the write action slow because the main memory and cache both needs to be updated
        while memry writes are pretty fast, the cpu might take some time or may be never to update the cache
        thus other threads that are accessing your varible might get late updates or never

        solution: <volatile> keyword does not cache the value of the variable and always read the variable from the main memory
        guarantees visibility and ordering

STRING
	sequence of characters
	internally just a character array
	immutable (since arrays are immutable i.e fixed size)
	so when a change is made, a new string instance is created

STRING BUFFER
	mutable strings
	has synchronized methods
	therefore thread safe
	therefore slower comparitively

STRING BUILDER
	mutable strings
	has non synchronized methods
	therefore are not thread safe
	therefore faster comparitively

GIT

	source code manament system (SCM) or version control system (VCS)
	maintains history of every change
	developers can collabratively work on same source code
	used for tracking code changes and who made those changes

	terms:
		repository
			working directory or project folder
			local and remote
		branches
			used to create another line of development
			default is master branch
			aka subversion
			can be different working or final versions
		commits
			a point in time state of the repo
		clone
			a copy repo either on local or another remote
		pull
			copies changes from remote to local
		push
			copies changes from local to remote repo
		head
			pointer that always points to the latest commit
		revision
			version of code
			generally represented by commits

	Typical Workflow:
		git branch or git clone or
		edit/code

```
        stage changes // git add .
        commit to local repository // git commit -m "message"
        push to remote repository // git push



    Set Up
        install git
            https://git-scm.com/downloads

        set username and email to tell git who is making changes
            git config --global user.name "java.june23@.gmailcom"
            git config --global user.email "java.june23@.gmailcom"

        sign up on github
        create a public repo

    Use
        clone remote repo on local
            git clone https://github.com/pavanpwm/sample-public-repo.git

        code
            add some files
            write code

        check changes
            git status // shows intracked files and code changes

        stage changes
            git add filename // tell git to trak changes in this file
            git add -A // stage all changes

        commit changes
            git commit -m "commit message here"

        view commit history
            git log



    Other Common Commands
        git switch _branch// switch to another branch
        git checkout _branch// create and switch to another branch
        git clone _repo// create a copy of repo
        git diff // show changes between commits
        git fetch // to check if there are any changes on remote repo
        git merge _branch// merge copy changes from remote repo
        git stash // stash changes away
        git stash pop // bring back the the stashed changes
        git revert _commit-hash // essentially creates a new poin copy of commit
        git reset _commit-hash // points the head to that commit



------------------------------------------------------------------------


INTERVIEW PREP
```

JAVA PROGRAMS

- Basic Programs
- Conversion Programs

- Class and Object Programs
- Method Programs

- Searching Programs
- 1-D Array Programs
- 2-D Array Programs

- String Programs
- List Programs
- Date and Time Programs

- Exceptions and Errors Programs

- Collections Programs

- Multithreading Programs

practice here:
https://www.geeksforgeeks.org/java-programming-examples/

JAVA

https://www.javatpoint.com/corejava-interview-questions

DATA STRUCTURES

- array
- stack
- map
- linked list
- queue
- heap

- tree
- graph

practise here:
https://practice.geeksforgeeks.org/explore?page=1&sprint=a663236c31453b969852f9ea22507634&difficulty[]=-2&difficulty[]=-1&difficulty[]=0&difficulty[]=1&difficulty[]=2&sortBy=submissions&sprint_name=SDE%20Sheet&utm_medium=main_header&utm_campaign=practice_header

---

ALGORITHMS

sorting
bubble, merge
searching
linear, binary
recursion

practise here:
https://practice.geeksforgeeks.org/explore?page=1&sprint=a663236c31453b969852f9ea22507634&difficulty[]=-2&difficulty[]=-1&difficulty[]=0&difficulty[]=1&difficulty[]=2&sortBy=submissions&sprint_name=SDE%20Sheet&utm_medium=main_header&utm_campaign=practice_header

---

SQL

https://www.javatpoint.com/sql-interview-questions

GIT

https://www.javatpoint.com/git-interview-questions

SPRING FRAMEWORK

https://www.javatpoint.com/spring-interview-questions
https://www.javatpoint.com/spring-quiz

HTTP

https://www.javatpoint.com/http-tutorial

REST APIs

https://www.interviewbit.com/rest-api-interview-questions/

SPRING BOOT

https://www.javatpoint.com/spring-boot-interview-questions

Basic HTML, CSS, JS, AJAX/FETCH
