Stage 1 - Core Java & advance java

Stage 2 - Selenium (4 hours (web automation) & AutoIT (windows automation)

Stage 3 - BDD framework

Stage 4 - Git and Jenkins (CI/CD pipeline)

Github

Java - <https://github.com/balaji-githubstore/JavaConceptHealthAsyst.git>

1. Java

* Language developer by Sun microsystems
* High level programming
* Platform independent
  + Write once run anywhere

Java Architecture

Source code (.java) 🡪 Byte code (.class) 🡪 O/P

1. JVM - Java Virtual Machine

* Source code - can be written in simple text file (.java)
* Java compiler - converting source code (.java) to byte code (.class)
* Byte code interpreted by JVM
* JVM varies from platform to platform or JVM is platform dependent

1. Installation
   1. JDK 8 (Java Development Kit)
      * By default JRE (Java Runtime Environment) will be installed

<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>

* 1. IDE - Eclipse/Intellij/netbean

Option 1-

* Download eclipse.exe
* Choose - Eclipse IDE for Java developer
* Install it

Option 2-

* Download installed package as zip folder.

<https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2021-12/R/eclipse-java-2021-12-R-win32-x86_64.zip>

1. Compile time - Source code to byte code

Run time - Bytecode to O/P

1. UpperCamelCase - MyFirstProject

lowerCamelCase - myFirstProject

1. Eclipse Structure

Workspace - lowercase

Project 1 - UpperCamelCase

Package 1 - lowercase (com.healthasyst.purpose) or (org. healthasyst.purpose)

Class 1 - UpperCamelCase

Methods/variables - lowerCamelCase

Class 2

Methods/variables

Package 2

Project 2

1. Datatypes
   1. Primitive datatype/pre-defined datatype
      * Byte - 8 bits

1 bit - sign (+ or -)

7 bit - storing the number

* 1. Non-Primitive datatypes/ Non - predefined datatypes - collection of primitive datatypes
* String
* Array
  + Size should be defined
  + Zero based index

1. Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 6

at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)

at java.base/java.lang.String.charAt(String.java:712)

at com.healthasyst.datatypes.DatatypesDemo.main(DatatypesDemo.java:44)

1. Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5

at com.healthasyst.datatypes.ArrayDemo.main(ArrayDemo.java:14)

1. Debugging

* Resume
* Terminate
* Step into
* Step over

1. Conditional statement
   1. If condition
   2. Switch condition
2. Relational operators

==, !=, >,<,>=,<=

1. Logical operator

And (&&), Or (||), not

1. Iterative statements
   1. For loop
      * Initial condition
      * End condition
      * Iteration
   2. For each loop - designed for handling the array and collections
   3. While
   4. Do while - minimum once it will execute
2. break - to stop the further iteration

Continue - just moves to next iteration

1. **Methods - It’s a building block of your program**

* Reusability
* Maintenance
  1. **Create and call static method**

//accessmodifier static returntype methodname(arguments)

**public** **static** **double** areaOfCircle(**int** r)

{

**double** area = 3.14 \* r \* r;

**return** area;

}

**To call the static method** - **classname.methodname(args)**

* 1. **Create and call non-static method**

//accessmodifier returntype methodname(arguments)

**public** **double** areaOfTriangle(**double** base, **double** height) {

**return** (base \* height) / 2;

}

**To call the non-static method** -

* **Create object**
* **Use *objref.methodname()***

1. **Object** 
   1. **Declaration -** Area obj
   2. **Instantiation - new**
   3. **Initialize -** Area()
2. **Variable** 
   1. **Create and call static variable**
   2. **Create and call non-static variable**

1. Static vs Non-static with variable
2. Class & Object
3. Access modifier
   1. Private - accessible within the class
   2. Default - accessible within the package
   3. Protected - - accessible within the package and also in the inherited class
   4. Public - anywhere
4. SOLID Principle

S-Single Responsibility Principle - A class should have one and only reason to change, it should have only one job.

1. Exception in thread "main" java.lang.NullPointerException: Cannot read field "aNS" because "obj2" is null

at com.healthasyst.variable.VariableDemo.main(VariableDemo.java:32)

1. **this** keyword can be used under any method to refer the current object.
2. Stack and heap
3. String - immutable
4. Git - Git is a free and open source **distributed version control system.**

Download and install - <https://git-scm.com/>

**Architecture**

Project (local system) 🡪 local repository (local system) 🡪 remote repository (github, AWS code commit, Bit Bucket)

Steps

Local repo

1. git init 🡪 initialize the local git repo
2. git add README.md 🡪 collect the files,folder ready to move
3. git commit -m "first commit" 🡪 update the local repo

Server repo/remote repo

1. git remote add origin <https://github.com/balaji-githubstore/JavaConceptHealthAsyst.git> 🡪 register the remote url with the name origin
2. git push -u origin main
3. Constructor

* Pre-requisite for the object
* Constructor name and class name should be same. There is no return type.
* There will be default constructor which helps to load all the non-static (instance) variable with default value.
* We can override the default constructor by creating the constructor explicitly
  + Constructor without arguments
  + Constructor with arguments
* Whenever constructor created then we need to call that constructor on creating the object.

1. Static Polymorphism/ Compile time polymorphism/ early binding - Method to be called is resolved during compile time.
2. Constructor Overloading

By change in

* 1. Number of parameters/arguments
  2. Sequence of parameters
  3. Datatypes of parameters

1. Method Overloading

Multiple method with same name

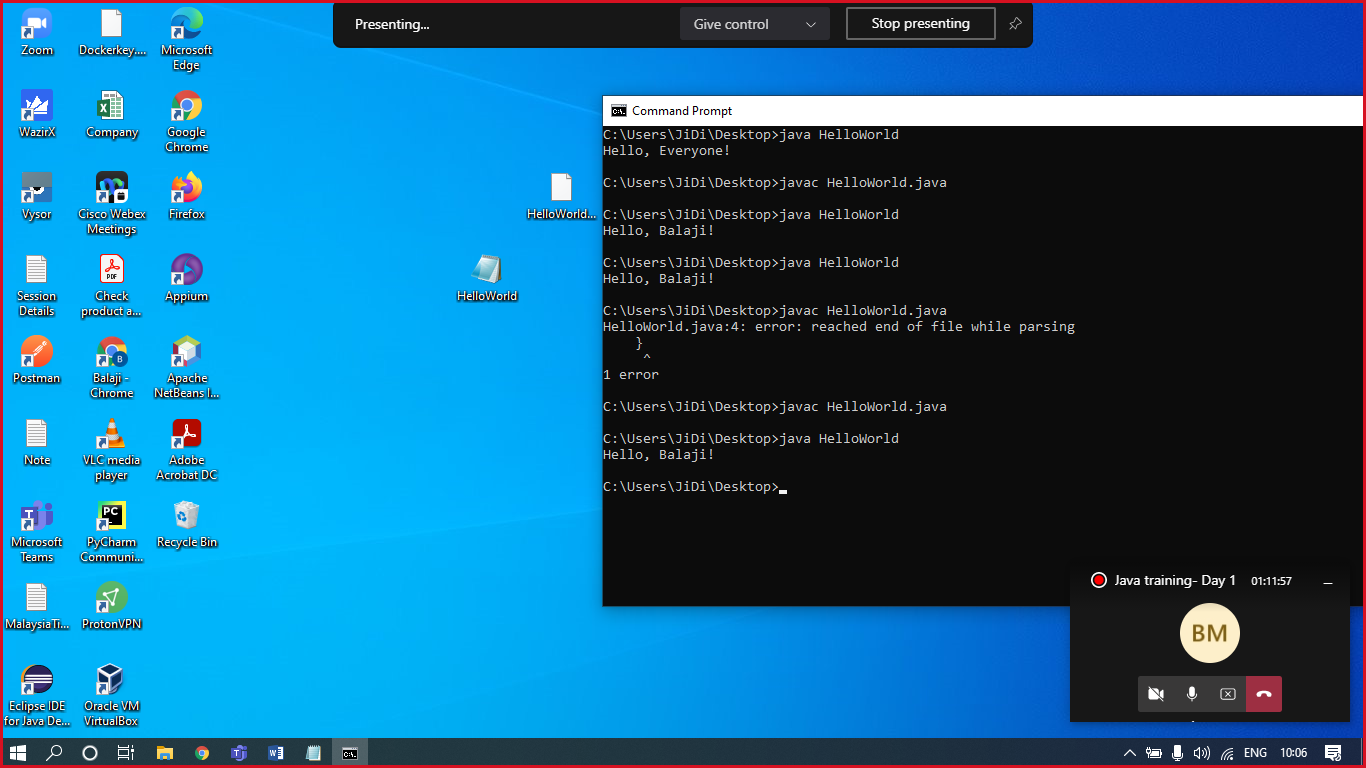
By change in

* 1. Number of parameters/arguments
  2. Datatypes of parameters
  3. Sequence of parameters

1. Inheritance - reusability
   1. When “is-a” relationship existing between two classed, we use inheritance

Reference:

Ref1: Img 1:



Ref2:

1GB - 1024 MB

1 MB - 1024 KB

1KB - 1024 B

1 B - 8 bits

1 bit (either 0 and 1)

Ref3:

<https://edabit.com/>

Task 1:

|  |  |
| --- | --- |
| **PERCENTAGE** | **GRADE** |
| 90 and above | A |
| 80 to 89 | B |
| 60 to 79 | C |
| below 60 | F |

**int** i=1;

i++; // or i++ // i=i+1

System.***out***.println(i);

Template for Student

Git:

