**JAVA BASICS**

.java(source) -> jdk(jre/jvm) – compilation

.class(bytecode) – compiler generated file – not human understandable

JVM (Java Virtual Machine)

* No physical existence
* Abstract in nature
* Loads the code, Verifies the code, executes the code -> Runtime Env
* Platform dependent

JRE (Java Runtime Environment)

* Physically exists
* It is an implementation of JVM
* Platform dependent
* Minimum requirement for executing any java code

JDK (Java Development Kit)

* It is a full featured standard development kit
* JRE + Developer Tools
* To develop and execute java code, minimum requirement is JDK

Compilation – is always with file extension (Ex: javac HelloWorld.java)

Execution – is always without file extension (Ex: java HelloWorld)

Class name ===== File name

CLASS -

* Blueprint from which individual objects are created
* Defines a new datatype
* Combination of properties(members) and behavior(methods)

public class SmartPhone {

storage,screensize,model,manufacturer -> properties/members

talk(),clickpic(),recording() -> functions/methods

}

OBJECTS –

* Instances of class

Iphone – storage,screensize,model, manufacturer

Samsung s3 - storage,screensize,model, manufacturer

Nokia - storage,screensize,model, manufacturer

Redmi - storage,screensize,model, manufacturer

Class specific - Members/Properties

5 pdf files – 2 Technical / 3 nontechnical -> 1 English 1 Tamil 1 Kannada

3 docx

3 mp3

6 txt

5 html

10 java

MAINTAINENCE

100 java files – use sub-folders to place these files

FOLDER / PACKAGE

com.company\_name.client\_name.project\_name.module\_name.submodule\_name;

Eclipse – Tool/IDE(Integrated Development Environment)

Workspace – physical folder location

Perspective – collection of different views(default – Java)

Views – different portions/tabs of the perspective

Create Java Project – name – jdk1.8 - finish

Right click on src folder - > Specify the package first and then the class name -> next -> finish

Java Project gets created on the left-side(Package Explorer)

Class/Interface

API?

Interface – User n Program, enables interaction between different components in the application

video

Java API – go through the docs

<https://docs.oracle.com/javase/7/docs/api/>

HISTORY

James Gosling – early 1990’s – Oak (Tree)

Java1.0 May 20,1995, Sun Microsystems

.

.

.jdk1.5 – Java 5.0

.

Jdk1.8 – Java8.0 (stable)

Java – statically typed language – int x=10;

char c=’h’;

String str=”string”;

Datatypes – Primitive(8) and Non-Primitive

Primitive(8) –numeric - int, double, float,long, byte, short

non-numeric- char,boolean

Non-Primitive – String, Arrays, User-defined classes

Primitive – Stack memory

Non-Primitive – Heap memory

Identifier/Variable

int number1=10;

camel-case

Type Conversions –

1. Automatic/Widening

Byte -> Short -> Int -> Long -> Float -> Double

1. Type-casting/Narrowing

Double -> Float -> Long -> Int -> Short -> Byte

CLA (Command Line Arguments)

main(**String args[**])

args[0]=1

args[1]=10

args[2]=100

Wrapper Classes –

int – Integer Integer.parseInt(String) – string to int

Integer.valueOf(String) – string to int

double – Double

char – Char

Autoboxing - primitive to non-primitive(Object)/Wrapper Classes

and Unboxing – non-primitive to primitive

Assignment – 20/05/2021

1. Think of different classes and objects in real-time and list it down
2. Attach source files to Eclipse
3. Understand stack and heap memory wrt datatypes
4. Go through Java API documentation
5. Java Coding Conventions – do a complete reading and follow it
6. Develop a program for playing guessing game.
7. Demonstrate Autoboxing and Unboxing through an example program.

Packages

1. Built-in packages
2. User-defined packages

Import – visibility

First stmt in the source file just after package

Java.lang – default package

Java.lang

Java.lang.instrument

Static import

Control+Shift+o

Operators –

Arithmetic -> +,-,\*,/, %

Relational -> <,>,<=,>=,==,!=

Assignment -> =

Ternary -> ?: (if-else)

Bitwise -> &, |, ^

Logical -> &&, ||, !

Unary -> +, -,++, --

Shift -> <<,>>, >>> (zero extension/strict)

Roots of quadratic equation:

Short-circuit AND and Short-circuit OR

Compound Assignment Operators/ Short-hand assignment opertors

a+=4 ================🡺 a = a+4

Comments – Single-line and multi-line

Precedence and Associativity - http://www.cs.bilkent.edu.tr/~guvenir/courses/CS101/op\_precedence.html

Program Control Statements

1. **Selection** – if, if-else,nested if, switch
2. **Iteration**- for, while, do-while
3. **Jump** – break, continue, return

Reading the input through Scanner class(different kinds of input)

java.util

Variables – local,instance,reference

Instance Variable – Every object will contain its own copy of variables/properties called Instance Variable

Local – Method, block{ }, constructor

Reference – is used to point to object/values (dot)

Ex: Vehicle car1 = new Vehicle();

Vehicle car2 = car1;

Methods – access\_Specifier returntype name\_of\_method(parameters) {

}

**Access Specifiers:**

Default

Private

Protected

Public

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | default | Private | protected | Public |
| Same class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

**Constructors(No-arg/Parameterized)**

1. A constructor initializes an object when it is created
2. It has the same name as class
3. Syntax is similar to a method
4. Constructors have no return type
5. CANNOT be abstract, static, final and synchronized

This keyword, Overloading

this() -> calling default constructor, it has to be the first line

* It can be used to refer to the instance variables of current class
* It can be used to invoke current class/default constructor
* It can be passed as an argument in method call
* It can be passed as an argument in constructor call

Method Overloading – (Static Polymorphism)

1. Vary the number of parameters
2. Vary the types of parameters
3. Order of parameters

Garbage Collection –

Free up heap memory

Automatically done by JVM

1. There are objects to be garbage collected
2. There is need a heap memory

Garbage collector –

1. System.gc()
2. Runtime.getRuntime().gc()

finalize() – Object

Island of Isolation

final,finally,**finalize**

Assignment – 21/05/2021

1. Use static import to find the roots of quadratic equation

(Hint: use java.lang.Math)

1. >>> - understand the working wrt MSB
2. Understand ternary operator with an example program
3. Construct truthtable.
4. Constructor Overloading – Create a class called Box

(width,height,depth) volume()

Read the input from the user

1. Island of Isolation – What is this all about
2. JIT – Just-in-time Compiler – self learning
3. Determine the value of the following expressions-

(a=5,b=2,c=1)

a) m=++a\*5;

b) n=b++-c\*2;

(x=10, y=2,p=2)

c) p\*=x/y;

9. Switch case for displaying syntax for all Program Control Structures (Nested Switch)

**Object class** (java.lang)

Understand all the methods of Object class through documentation

**Abstract class**

* An abstract class must be declared with an “abstract” keyword
* It can have Abstract and non-abstract methods also
* It cannot be instantiated
* It can have a constructor and static methods

**Interfaces**

* Declares a set of methods that will be implemented by a class
* Are declared with interface keyword
* Provides Abstraction
* Any number of classes can implement an interface
* One class can implement any number of interfaces
* One interface can extend more than more interface also

**Static – Variable, Method, Block {…..} –** initialize static members**, Nested Class**

Ex: class Employee {

int empid;

String empname;

String company; //common property for all objects

}

Use – saves memory

Nested Classes –

1. Non-static Nested Classes
2. Inner Classes
3. Method local Inner Classes
4. Anonymous Inner Classes
5. Static Nested Classes

**Inheritance** –

Hierarchical classifications

SuperClass/Parent/Base

SubClass/Child/Derived

Advantages – Reusability, Extensibility, Data Hiding, Overriding

Types:

1. Single
2. Hierarchical
3. Multi-Level
4. Hybrid
5. Multiple

TwoDShape

Triangle **extends** TwoDShape

Case 1 – members have no access specifier(default)

Case 2 – public/protected

Case 3 – private

**super** keyword – variable, constructor, method

Method Overriding (Dynamic Polymorphism)

1. The method must have same name as that of parent class
2. The method must have same parameters as that of parent class
3. There must be a IS-A relationship

Assignment 24/05/2021

1. Explore
2. Method local Inner Classes
3. Anonymous Inner Classes
4. Static Nested Classes
5. Outer - CPU – Inner - RAM, Processor

Assume – different members and methods

1. Understand Diamond problem wrt Inheritance
2. Read the input from the user and generate its cube of that number (till that number starting from 1)
3. Display sum of odd natural numbers. Read the limit from the user
4. Generate a pyramid as shown in the class
5. Generate Flyod’s triangle and Pascal’s triangle
6. WAP – that accepts three numbers from the user and prints “increasing” if the numbers are in increasing order, “decreasing” if the numbers are in “decreasing order” and “neither increasing nor decreasing” otherwise

1234

5321

6759

**final** keyword

* variable (constant variable/named constants)
* method (preventing method overriding)
* class (prevents inheritance, create immutable class)

Arrays –

Collection of similar data elements

Referred by a single name/common name

Stored in continuous memory

Multiple values of the same type are stored in a variable

Index-based(starts from zero)

One-dimensional Array/ Multi-dimensional Array(Jagged Array)

Syntax: datatype[] variable\_name;

Ex: int counter[] = new int[3];

int[] counter = new int[3];

int[] counter= {1,2,3};

char[][] table=new char[3][4];

**For-each style/Enhanced for loop**

for(type itr\_var: arrayname)

int[] numbers={1,2,3,4,5,6,7,8,9,10}

int sum=0;

for(int i=0;i<10;i++) ======= for(int x: numbers)

sum=sum+numbers[i];

**String - “”**

Literal

String str=”JAVA”;(String Pool)/ String str = new String();(Heap Memory)

Char ch=’A’;

equals() and ==

Assignment 25/05/2021

1. Insert an element at a specific position in an array
2. Remove an element from a specific position from an array
3. Reverse the array elements
4. Sort the array elements (int and string)
5. Find duplicates in an array
6. Find element pairs which sum upto 10 (15 elements)
7. Find second smallest element and second largest elements in a specific array.
8. Add two matrices of same size
9. Multiply two matrices of same size (pXq qXr)
10. Check if an element is present or not
11. Try array reference (num1 and num2 10 elements, num2=num1, num2[3]=99)
12. Device some programs to understand jagged arrays –

Riders on weekdays / Riders on weekends

No. of riders and no. of trips(kms)

Rider? Yes -> Which day did u drive? Weekday/Weekend -> how many kms

1. Read about StringPool and Heap
2. Prove that String is immutable
3. Find difference between equals() and == wrt String
4. WAP to test if a given string contains the specified sequence of character values.
5. Develop a program to encode and decode as discussed

A – Z 0 – 25 Assume: space character – 99etc.,.

“The quick brown fox jumps over the lazy dog”

Any input should be encoded and decoded

String operations –

concat(), indexOf(), length(), charAt(), substring(), equals(), compareTo(), contains(), equalsIgnoreCase(), toString(), subsquence(), lastIndexOf(), regionMatches(), startwith(), endswith(), toUppercase(), to LowerCase(), contentEquals(), append(), replace(),insert(), split(), trim()

Features of Java – Secured, OO, Portable, Platform-independent, Architecture-neutral, Supports Multi-threading, Faster (JIT), Distributed, Robust, Stable, Dynamic, High Performance, Interpreted..

Exception– Recoverable, checked/unchecked, programs, java.lang.Exception

Error– irrecoverable, human-error, java.lang.Error

Ex: OutofMemoryError, IOError

Checked Exceptions/Compiletime Exception –

Classes which directly inherit Throwable class except RuntimeException and Error

Ex: IOException, SQLException,…….

Examples of Unchecked Exceptions/RuntimeException –

Classes which inherit Runtime Exception. They are checked at runtime/execution time and not checked during compilation

ArrayIndexOutofBoundsException

ArithmeticException

NullPointerException

Five keywords -

**try**

**catch**

**throw**

**throws**

**finally**

Exception caught in the catch block must be of the exact same type, otherwise, generally just catch Exception type

One try block can have multiple catch blocks

If you want to catch exceptions of both superclass as well as subclass types, put the subclass first in the catch sequence.

Order of catch clause is important.

Nested try – One try block can be inside another try block

try {

try{

} catch(){ //for inner try

}

} catch() { //for outer try

}

throw – manually throw an exception

**throw new exceptionObject;**

**throws – method**

access\_specifier return\_type methodName(parameters\_list) **throws Exceptions** {

}

Checked Exception can also be propagated with Throws

|  |  |
| --- | --- |
| throw | throws |
| Throw is used inside a method | Throws is used along with method signature |
| Checked exceptions cannot be propagated | Checked exceptions can be propagated |
| You cannot throw multiple exceptions | You can declare multiple exceptions |
| Used to manually throw an exception | Used to declare/predict an exception |
| Is followed by new keyword which is followed by Exception class | Is not followed by new keyword but is followed by Exception class |

Custom Exception Class –

extends Exception/RuntimeException

Assignment – 26/05/2021

1. Develop an application for Student registration.

Collect necessary details of the student.

If the student age is less than 12 and weight is < 35 kilos, student is not eligible for registration

1. Explore **rethrow** (One catch can manually throw an exception which can be handled by another catch block)
2. Bank Transaction application which has credit() – increase balance, debit() – decrease balance and get/read transaction history(). Use MinBalanceCustomException class to convey to the user that min balance should be maintained at 1000 rs and to do debit() transaction there should be more than minimum balance amount. NoNegativeException class which doesn’t allow negative amount to be entered by user for debit(Assumptions permitted)
3. Try NonIntResultException on the example program shown in the class.
4. try\_with\_resource

Interfaces –

1. Marker Interface – no methods
2. Functional Interface – SAM (Lambda Expressions)
3. Non-Functional Interface

**Multi-threading**

MS-word – type, spell-check

Light-weight process

Single-path of execution in a process is termed as thread

Thread is a unit of process

1. Extending Thread class
2. Implementing Runnable interface

Thread ID (unique), Name

Thread Priority

MIN - 1

MAX - 10

NORM – 5 (default)

start() and run()

start() – **new Thread is created** and code runs inside run()

run() – **no new Thread is created** and code inside run() on the current Thread.

Thread lifecycle

1. New 2) Runnable 3) Running 4) Wait 5) Terminate

Inter-Thread Communication (Object)

1. wait()
2. notify()
3. notifyAll()

Synchronization

**Synchronized – critical section (objects lock)**

**Methods BL**

**Block of code – synchronized(object) { …. }**

Assignment – 27/05/2021

1. Understand Serialization and Deserialization
2. Multi-threading – Producer-Consumer problem
3. ThreadPool, ThreadGroup, Deadlock
4. Suspending, Resuming and stopping of threads.
5. File – creation, reading, writing, appending

Properties file

Csv file

1. Withdraw() and deposit() – Customer – Thread Synchronization

Java Collection Framework

Array –

int data[] = new int[10];

data[11]=100;

* Static memory allocation/pre-defined size
* Homogeneous data

Why array length is fixed?

Program -? RAM

Disadvantages of Arrays –

1. Cannot change the length
2. No proper methods to add,remove or read the data

ArrayList

* Dynamic storage allocation
* There are proper methods to manipulate ArrayList.
* Heterogeneous data

Collection Framework (java.util.Collections)

Framework – Architecture for storing and manipulating the data

Collections –

List – ArrayList(not thread-safe), LinkedList(not thread-safe), Vector(thread-safe)

Set – HashSet,, TreeSet, LinkedHashSet

Map – HashMap(based on HashTable), TreeMap, LinkedHashMap

**List: (Ex: Shopping list)**

1. Used to store a list of objects.
2. Duplicates are allowed
3. Objects remain in the order (maintains the insertion order)
4. Elements are indexed via integer value
5. Checking for a particular item in a list is slow
6. Looking up an item by index is faster
7. Iterating through the list is fast
8. Sorting is possible

equals() -> checks if two objects are equal or not/content

== ->checks if two objects hold the same reference/memory location or not

**Set:**

1. Stores only unique values(No duplicates allowed)
2. It is not indexed
3. Fast to check if a particular element/object exists
4. Implement hashCode() and equals() [If you use user-defined datatypes]
5. Only one null value is allowed (only for HashSet)

Set and Map -> Concurrent Collection Framework

ConcurrentModificationException -> when performing modifications on the collection object which is currently in use

**Map:**

1. Key-value pairs
2. Like a lookup table
3. Retrieving a value based on the key is faster
4. Iterating over the Map is slow
5. Implement hashCode() and equals() [If you use user-defined datatypes]
6. Not Optimized
7. Not index-based but key-based

Assignment – 28/05/2021

1. Find how to sort TreeSet in descending order - possible or not?
2. List – understand how Vector works
3. Set – understand how LinkedHashSet works
4. Differentiate between AL and Vector
5. Create a student class with id,name,marks. In Student class, add PC and toString() methods, override hashCode() and equals(). Create Student HashSet and create five student objects and add it to Student Hashset. Try sorting based on various data members (use Comparable and Comparator)
6. Try LinkedHashMap