# OOAD

## Java



### Compilation Process

javac

→ class file → loaded into Classloader → check for bytecode → interpreter

→ execute

→ send it to the

CPU/hardware

class file = bytecode

### terminologies

JVM

JDK

software kit

JRE

Java Runtime Environment

JVM

### program I/O

System.out.println()

Scanner(System.in )

nextInt()

nextLong()

…

nextLine() string

## IDEs

IntelliJ

VScode

Eclipse

Visual Studio

### Online env:

replit

### Java as a language

Statically typed Language

Strongly Typed Language

### keywords

boolean byte char

false true

int short long

float double

## Data types

### Primitive Data Types

int 4

short 2

long 8

float 4

double 8

char 2

boolean 1 bit

byte 1 byte

float

12.36

1236 x 10-2

### Non Primitive Data Types

Reference Types

Arrays

Strings

Class

Objects

Interfaces

## Operators

### arithmetic

+

-

\*

/

% modulo

### assignment

=

compound

+= a += b

a = a + b

-= a = a -b

…..

### relational

<

>

<=

>=

==

!=

### logical

&& and

|| or

! not

### unary

-

a = -100

++

post

pre

### bitwise

& and

| or

^ ex-or

### shift

<<

>>

>>>

<<

0011 1100 x

0111 1000 y

0001 1110 z

0011 1100 x

1111 0000 x << 2

0000 1111

+1

0001 0000

1010 0000

1010 0000 0000 0001

0001 1111 31

0000 0011 3

0000 0100 -4

### terniray

## Flow control

### if else

if

else if

else

### switch

switch

case

break

default

### while

while

do while

break

continue

### for

for

break

continue

### for-each

loops automatically

assigns too

features:

no index

can not be effectively used to change contents of the array

moves only forward

moves only in single steps

## Arrays

* dynamically allocated
* continuous memory allocation
* objects
* [ ]

int [ ] arr = {10, 20, 30}; //length is 3

arr.length → data

String stra = “atlas” // length is 5

stra = “amazon” // length is 6

stra.length() → function

## Strings

### String

* objects
* immutable

String literal

* String Constant pool
* JVM optimize

new operator

* dynamically allocated
* heap

### String methods

str.length()

str.toUpperCase()

str.toLowerCase()

str.indexOf(“”) the index of that particular sub-string

str.charAt() the character the specified index

str.isEmpty()

### StringBuffer

a t l a s t

01 23 4

### StringBuilder

faster than StringBuffer

not thread-safe

## OOPs

### general concepts

Object class is a superclass

#### vocabulary

state

(values)

properties

behaviour

methods

identity

class (memory) division

objw objx objy



#### class declarations, in general:

modifiers

public

private

constructor

methods

#### declaring member variables

various kinds:

within a class → fields

within a method (or block of code) → local data

in method declaration → parameters

#### method name (conventions)

speak

speakLoudly

getData

runFast

changeRoomTemperature

#### objects

Bank objy = new Bank(234682, 3000, "savings");

Bank objy → declaration

new → instantiate

Bank(xxx, yy) → initialisation

#### garbage collection

JRE deletes objects when it feels that those objects are no longer being used

runs automatically

an object can be deleted, when:

-- no references to that object

#### Features

Encapsulation

Inheritance

Polymorphism

function/methods overloading

C++ Java

operator

C++ Python

Abstraction

## Polymorphism

4 + 6

“hello” + varx

Static

compile time

go\_for\_dinner(Saturday)

Runtime

Run time

Function Overloading

* num of parameters
* data types of parameters
* order of parameters

## Encapsulation

capsule

hide the data

## Object & Class Relationships

Association

Composition

Aggregation

### Association



association manages:

one-to-one

one-to-many

many-to-many

two forms of association:

composition

aggregation

# Data Structures (Java)

## types of DS

primitive

byte, short, int ….

non primitive

1. Linear Data Structure
   1. Array
   2. Linked List
   3. Stack
   4. Queue

…..

1. Non-Linear Data Structure
   1. Tree
   2. Graph

## Array

linear data structure

collection of data/variables/objects

int, float

Strings

objects

referenced by a common name

contiguous memory location

data type = base type

Arrays are considered as objects

heap area (dynamic memory)

new

size of array = int

length

public final length

Object class is a superclass of array

Linear Data Structure

advantages:

random access of elements (using index)

access is faster

sorting, iteration is easier

modification (of values )

disadvantages:

size is fixed

modification (insertion, deletion) cumbersome

time taking

capacity is huge, used section is less, then memory is wasted

always needs contiguous memory locations

for(int val : arr)

{

// body of the loop

}

### clone

#### while passing to a function:

arr

| 10 | 20 | 30 | 40 | 50 | 60 |
| --- | --- | --- | --- | --- | --- |

brr = arr

brr was referencing back to arr

crr = arr

crr was referencing back to arr

any changes done to crr, effects arr too

#### while passing to a function by clone, or by equating using cloning:

arr

| 10 | 20 | 30 | 40 | 50 | 60 |
| --- | --- | --- | --- | --- | --- |

brr = arr.clone()

brr is a new copied version of arr

separate area in the memory is created

crr = arr.clone()

crr is a new copied version of arr

any changes done to crr, does not effect arr

crr

| 6 | 16 | 26 | 36 | 46 | 56 |
| --- | --- | --- | --- | --- | --- |

### multi dimensional arrays

arr[m][n]

m rows

n columns

m number of n arrays

arr [6][4]

6 one-dimensional arrays

each one dimensional array is of the size 4

ways of creating multi dimensional array:

int arr [ ] [ ] = new int [6] [4]

int [ ] [ ] arr = new int [6] [4]

## Linked List

tiffin box mould

start = 7000

| 8055 |
| --- |
| chole bhature |
| 7054 |

7000

| 7000 |
| --- |
| momos |
| 8000 |

7054

| 7054 |
| --- |
| pasta (red sauce) |
| 9011 |

8000

| 9011 |
| --- |
| shawarma |
| 8055 |

8020

| 8020 |
| --- |
| biryani |
| 7000 |

8055

| 8000 |
| --- |
| chicken masala |
| 8020 |

9011