JAVA

Topics covered:   
1. **Introduction**  
2. **Installing of JAVA**  
3. **Java Architecture**  
4. **Java Programming Format**  
5. **Java Language Fundamentals**1. **Tokens**  
1.1 **Identifiers**  
1.2. **Literals**  
1.3. **Keywords**  
1.4. **Operators**  
2.**Data types**  
3.**Typecasting**  
4. **Java statement**  
5.**Arrays**  
6. **Variables**  
7. **Control statements**   
7.1**Select statements:**  
**if**  
**Nested ifs**  
**if else if ladder**  
**switch**  
**7.2Jump statements:  
break  
continue  
return**  
**7.3iterator statements:  
for  
while  
dowhile  
8.Methods**  
**9.OOPS   
Classes and objects  
super keyword**  
**Object class**  
**10. Constructor**

**11.Inheritance**  
**12.Abstraction   
Abstract class**  
**Java types  
13.Interfaces**  
**14.Encapsulation**

**15.Polymorphism**  
**16.IIB  
17.SIB**  
**18. Exception handling**  
**19. Wrapper classes**  
20. Collections  
**List**  
Queue  
**Set**  
**Map**  
**21.Strings**

What is programming?  
- Programming is a way to “instruct the computer to perform various tasks”.

What is JAVA?  
\*JAVA is a high-level programming language. (Human understandable form - English)

\*JAVA is a object oriented programming language designed to develop all kinds of applications.

Programming paradigms available in market?  


Who invented java ?  
James Gosling – June 1991 - OAK  
Sun micro systems – 1996 – first version of java

Oracle – 2001 – 2025 (java is maintained)

What are the advantages of java?

1. Simple
2. Object oriented
3. Platform independent
4. Architectural neutral
5. Portable
6. Robust
7. Secure
8. Dynamic
9. Multi-threaded
10. Distributed
11. High performance
12. Interpretive
13. Simple

* Less memory and less execution time
* Removed all the confusion features like pointers, multiple inheritance, operator overloading
* Simplified syntax from c and c++

1. Object oriented

* Able to store data in the form of objects.

1. Platform independent

* Compile on one OS and to execute on another OS.

1. Architectural neutral

* One H/W arch and to execute on another H/W arch.

1. Portable

* Able to execute applications on all the os and all the H/W systems.

1. Robust

* Very good at Memory management system and having exception handling mechanisms
* Heap memory management system – allocates and deallocates memory of the objects at the run time.
* Very good predefined library

1. Secure

* JAAS [Java authentication and authorization service]
* Security manager inside the JVM – java virtual machine – implicit security
* JAAS to provide web security
* Network security algorithms

1. Dynamic

* Allow memory allocation for primitive data types at run time, not at compilation time.

1. Multi-threading
2. Distributed

* Standalone application
* Distributed application – web services, EJB’s, socket programming etc…

1. High performance
2. Interpretive

* Compilative and Interpretive

What are the dis advantages of java?

UI components

What is the full form of Java ?

* No full form
* Coffee symbol – coffee bean name

Diff technologies available in java ?

1. Java se – standard edition – desktop or standard applications
2. Java ee – enterprise edition – Websites for banking, health care etc..
3. Java me – micro edition – coffee machine, atm machine …

Explain **Java Architecture** ?  
Principle – WORA (Write once and run anywhere)





JDK – Java development kit

JRE – Java Run time environment

JVM – Java virtual machine

JDK = JRE + Development tools

JRE = JVM + Library classes  
  
JDK – Java development kit – write, compile, debug, and run java prog.

JRE – Minimum run env +   
Development tools :   
javac – java compiler – java source code(.java) -> Java byte code(.class)  
java – java application launcher  
jar – tool for creating and managing java archive file which helps in distribution of code.  
Javadoc – a documentation generator.  
  
**JVM – Platform independent**



* JVM is the Interpreter byte code – machine code (instructions)
* Platform independent
* Loads the code
* Verifies the code
* Executes the code
* Provides run time environment

\*\*\*  
JVM performs following key steps :

1. Class Loader : loads the .class files into the memory  
   This involves 3 phases:

1.1 Loading : Locates and loads the binary data of a class.  
1.2 Linking : following 3 stages   
 1.2.1 Verification : structural correctness and security violations

1.2.2 Preparation : memory for static variables and initialize them with default values  
 1.2.3 Resolution : replaces symbolic references in the byte code with direct references.  
 1.3 Initialization : executes static initializers and static blocks in the class

2. Java Run time date area / JVM memory area   
 The JVM manages various memory areas during execution :  
 2.1 Method area : Stores class – local data , including meta data, static variables, and some method code.  
 2.2 Heap area : The run time data area where objects and arrays are allocated. This is the place where garbage collection primarily operates.   
 2.3 Stack area : Each thread in the JVM has its own private stack. Stack frames are created for each method call, storing local variable’s, operand stack and method return values.  
 2.4 PC Registers : Each thread has a PC (program counter) register that stores the address of the currently executing JVM instruction.  
 2.5 Native method stacks : Used to support native methods

3. Execution engine : Executing the byte code

3.1 Interpreter : Interprets byte code instructions line by line and executes them. This will be usually slow.  
 3.2 JIT compiler: Just in time – improve performance, jit compiler compiles frequently executed byte code. Executing the byte code much faster.   
 3.3 Garbage collector : Automatically reclaims memory occupied by the objects that are no longer referenced / unused.   
  
4. Java Native interface (JNI) : java code to interact with native applications and library written in other prog languages.

Why we need to install Java software into our system ?  
How do we download java software ?  
https://www.oracle.com/in/java/technologies/downloads/  
Which version of java software need to be installed ?  
 *Long-Term Support (LTS)*  - JDK21  
Short-Term support(STS)  
<https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe>   
[jdk-21\_windows-x64\_bin.exe](https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe)   
How do we set the java path in our system ?  
1. Go to this PC > Right click > Properties or  
Go to Control Panel\System and Security\System  
2. Click on Advanced system settings   
3. Click on Environment variables  
4. Click on New   
Variable name : JAVA\_HOME  
Variable value : C:\Program Files\Java\jdk-22  
5. Go to Path variable   
Click on New   
%JAVA\_HOME%\bin   
How do we cross check which version of java installed in out laptops ?  
Go to the cmd prompt and type java --version  
  
Where we need to write the java program ?  
Note pad or Note pad ++

Tools available for writing java programs ?  
1. Eclipse  
2. Intellij IDEA  
3. Netbeans  
4. Visual studio code  
  
Open the note pad and type   
  
class Test{

public static void main(String[] args)

{

System.out.println("Welcome to nammaqa bng");

}

}  
  
Save – “Test.java”  
open the cmd prmt > javac Test.java > java Test  
  
Download and install eclipse:  
https://www.eclipse.org/downloads/   
  
Steps to create java project   
\* Select the default work space and start the eclipse   
\* For creating the java project you can click on create a java project directly or click on File > New > Java project  
\*Provide “JavaPractice” in project name.  
\*Click on Next   
\* Click on Finish  
\*create a package by right click on src folder and > New > Package > “com.day1” – package name > save  
\*Create a class by right click on package name > New > Class > “Prog1” – class name > select public static void main > Finish  
\* Right click on Prog1.java > Run As > Java application > To run the java files and see the o/p in the console.  
  
Package section :  
Package is collection of related classes and interfaces as a single unit.  
Following advantages :  
1. Modularity

2.Abstraction  
3. Security  
4.Sharability  
5.Reusability  
There are 2types of packages in java :  
1. Pre defined packages -   
Eg : java.io, java.util, java.awt, java.sql….  
2. User defined packages  
Eg: org.app.login, org.app.signup  
syntax : package package\_name;  
2 conditions for packages :  
1. Package declaration statement must be the first statement in java .java files.  
Q: is it possible to provide more then one package declaration with the same name ?  
No  
Q: Can I place 2 package names in my java files ?  
No  
2.Package names must be unique, they must not be sharable and they must not be duplicated.  
  
import section :  
\* is to make available all the classes and interfaces of a particular package into present java file.  
syntax : import package\_name.\*;  
Eg :   
**import** java.io.\*;  
**import** java.lang.\*;  
**import** day2.\*;  
\* **import** java.io.\*; -> Able to import all the classes and interfaces from java.io package  
\* **import** java.io.BufferedInputStream; -> Able to import only the specified member from the specified package.  
  
Note : 1 package statement but we can have multiple import statements.  
  
Q: To use the classes and interfaces of a particular package in java files is it mandatory to import that package ?  
No  
  
Eg : With import statement :   
**import** java.io.\*;  
BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

System.***out***.println("Please enter your name");

System.***out***.println("Your name is "+br.readLine());

Without import statement :   
java.io.BufferedReader br = **new** java.io.BufferedReader(**new** java.io.InputStreamReader(System.***in***));

System.***out***.println("Please enter your name");

System.***out***.println("Your name is "+br.readLine());

Java Programming Format :  
To design basic java application within single java file we have follow following structure:



Comment section :  
3 types :  
1. Single line comment :   
Syntax : //---------Description-------------  
2. Multi line comment :   
Syntax :   
/\*-----------  
--------------  
---Description--  
-------------  
--------------\*/  
3.Documentation comment:  
Syntax  
/\*  
\*------------------------  
\*------------------------  
\*--Description--------  
\*-----------------------  
\*/  
  
Class section :   
class - Blue print  
object – real world entity  
Syntax :  
**public** **class** className {

}  
  
Q: What are the diff members of a class?  
Variables and methods  
  
Main class section :  
contains main() method  
1. To define application logic in java application.  
2. To define starting and ending point for the application execution.  
Syntax  
**public** **static** **void** main(String[] args) {

}  
  
Execution of java program will follows following steps :  
\* On right click run as java application : jdk will helps in converting .java to .class file.  
\*jvm will search for .class file and look for main() method present or not  
\***If the required .class file is not available at the specified location, then jvm will throw :  
java.lang.NoclassDefFoundError.   
  
java.lang.ClassNotFoundException -> after compiling the code we have passed the wrong class name to execute the prog.**  
\*When jvm identifies main .class file at the specified location then it will load byte code into the memory, this phase is called as “class loading”, this would be performed by “class loader” component existed inside JVM.

\*After loading main class byte code to the memory, jvm will search for main() method, if main() method does not exists then jvm will throw :  
Main method not found in class Test, please define the main method as:

public static void main(String[] args)  
\*If main() method is identified in main class bytecode then jvm will create a thread to access main() method called as “Main thread”.

\*When main thread reached to main() method ending point then main thread will get dead state, with this jvm will stop all of its internal process and jvm will go to shut down mode.  
  


Language Fundamentals:   
1. Tokens  
2.Data types  
3.Typecasting  
4. Java statement  
5.Arrays  
  
1.Tokens  
Smallest logical units – “Lexeme”.  
Q: What are tokens in java?  
Eg: keyword – Token

|  |  |  |  |
| --- | --- | --- | --- |
| int | for | break | continue |
| lexeme | lexeme | lexeme | lexeme |

Operator – Token

|  |  |  |  |
| --- | --- | --- | --- |
| + | - | \* | / |
| lexeme | lexeme | lexeme | lexeme |

Types of tokens:  
1. Identifiers  
2. Literals  
3. Keywords  
4. Operators  
  
1. Identifiers   
name assigned – variable, methods, classes etc…  
Q What are the rules for creating identifiers?  
Rules   
1. Must not start with a number  
2.May start with an alphabet, \_ symbol, $symbol, but subsequent symbol must start with an alphabet, a number, \_ symbol and $ symbol.  
3. Not allowing spaces in the middle.  
4. Should not be duplicated  
5. All predefined class names  
  
**int** empNo = 10;  
  
int = lexeme  
empNo = Identifier  
= = operator  
; = special symbol/terminator  
  
  
Suggestions :   
1. Must be meaningfull  
  
 String xxx = "abc123"; //not suggestible

String accNo = "123"; //Suggestible

1. There is no length restriction  
     
   String Empxxx\_temp\_adress\_xxxxxxxxxxxxxxxxxx = "abc123"; //not suggestible

String EmpTempAddress = "123"; //Suggestible

1. Use (\_)  
   String EmpTempAddress = "123"; //not Suggestible

String Emp\_Tem\_Address = "123"; //Suggestible

2. Literals  
Literal is a constant assigned in the variables

**int** i = 10;

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

//int - lexeme, keyword, data type

//i - identifier

//= - operator

//10 - constant / literal

//; - terminator

1. Integral/integer literals
2. Floating point literals
3. Boolean literals
4. String literals

3.Keywords/Reserved words

Eg: goto, const..  
  
List of keywords:  
1. Data type and return types :  
byte, short, int, long, float, double, char, Boolean, void, return …  
2. Access modifiers :  
public, protected, private, static, final, abstract, native, synchronized, strictif ..  
3. Flow controllers :  
if, else, switch , case, default for, while, do, break, continue, return ….  
4. Class/object related :  
class, extends, interface, implements, enum, new, this, super, package, import …  
5. Exception handling:  
throw, throws, try, catch, finally…



4. Operators  
4.1 Arithmetic operators: +,-,/,\*,%,++,--  
4.2. Assignment operators: =, +=,-=,/=,%=  
4.3. Comparison operators: ==, != <, >, >=,<=  
4.4. Boolean logical operators: &, |, ^  
4.5. Bitwise logical operators: <<,>>  
4.6. short circuit operators: &&, ||



And - &

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

OR - |

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

2.Data types \*\*\*  
Q:What are the advantages of declaring data types?  
Following advantages:   
1. We are able to identify memory sizes to store data on the basis of data types.  
2. We are able to identify range values which we are going to assign to the variables on the basis of data types.  
Q:What are the diff data types available In java?

Following data types:   
1. Primitive data types / primary data types  
1.1 Numeric data types  
1.1.1 Integral/integer data types  
byte  
short  
int   
long   
1.1.2 Non integral data types /Decimal type  
float  
double  
  
1.2 Non numeric data types  
char  
Boolean

2. Non -Primitive data types / non - primary data types

2.1 Array  
2.2 Collections  
2.3 Interface  
2.4 Class

Q: What is data type?\*\*\*  
A data type is a classification of data which tells the computer or interpreter how the programmer intends to store/use the data.  
  
Computer memory units:  
bit = binary unit/digit  
1 Nibble = 4 bits  
1 Byte = 8 bits  
1 kilo byte (KB) = 1024 bytes  
1 Mega byte(MB) = 1024 kilobytes  
1 Giga byte(GB) = 1024 mega bytes

1 Tera byte(TB) = 1024 giga bytes  
1 Petabyte(PB) = 1024 tera bytes  
Q: What is wrapper class?  
Classes representation of all the primitive data types are called as “Wrapper classes”.  
\* since data types are used for declaration only we cannot perform any action directly on the data types. In order to perform any operation we use “Wrapper class” (pre-defined classes).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Wrapper classes | data types | Size | Range |
| Data types | Primitive | Number | Integral type | Byte | byte | 1 byte | "-128 to 127" |
| Short | short | 2 bytes | "-32768 to 32767" |
| Integer | int | 4 bytes | "-2147483648 to 2147483647" |
| Long | long | 8 bytes | "-9223372036854775808 to 9223372036854775807" |
| Decimal type | Float | float | 4 bytes | "1.4E-45 to 3.4028235E38" |
| Double | double | 8 bytes | "4.9E-324 to 1.7976931348623157E308" |
| Boolean | | | boolean | 1 bit |  |
| Character | | | char | 2 bytes | "-32768 to 32767" |
| No-primitive | Arrays | | | | |  |
| Collections | | | | |  |
| Interface | | | | |  |
| Class | | | | |  |

Q: What is the default value of integer data types in java ?

* Default value of primitive data types is 0. \*\*\*



3.Typecasting/type conversion  
Q: What is type casting/conversion?  
converting from one data type to another data type is called as type casting.  
Q: What are the diff types of type casting/conversion available in java?  
There are 2 types of type casting  
1. Implicit casting or implicit conversion or automatic conversion or widening or upcasting

2.Explicit casting or Manual conversion or Narrowing or Down casting

1. Implicit casting:  
The process of converting the data from lower data type to higher data type is called as implicit type casting.  
byte -> short -> int-> long -> float -> double  
\* There is no data loss in implicit casting  
jvm will perform following actions :  
1. Type casting : Coveting right side variable data to the left side variable data type implicitly.  
2. Value copy : Transferring value from right side variable to left side variable.

  
2. Explicit casting:  
The process of converting data from higher data type to lower data type is called as explicit typecasting.

To perform explicit casting we have to use following syntax :  
P a = (Q) b;  
Where ‘b’ variable data type should be higher than ‘P’  
Where ‘Q’ must be either same as ‘P’ or lower than ‘P’  
  
byte <- short <- int<- long <- float <- double  
  
\* There will be data loss in explicit casting  
  
Limitations of type casting/conversion:  
1. Data loss while doing the explicit conversation  
2. Boolean and char cannot be converted to any data type.  
3. When ever we are performing operations on lower data type with higher data type then collecting variable should be at least same as higher date type or still higher data type is prompted by JVM. This process is called as “Automatic type promotion”.  
\* byte to short operation will be promoted to int  
\* short to int operation will be promoted to int  
\* long to int operation will be promoted to long  
\* long to float operation will be promoted to float  
\* float to double operation will be promoted to double  
  
Q:What is automatic type promotion in java and give some examples?

4.Java statement:  
Statement is collection of expressions.  
1. Conditional statements:  
if, switch  
2. Iterative statements:  
for, while, do-while  
3. Transfer statements:  
break, continue  
4.Exception handling  
throw, try-catch, finally  
  
**1. Conditional statements:**  
These java statements are able to allow to execute a block of instructions under a particular condition.  
Q: What is a decision-making statement/conditional statement/control statements?  
A: Decision making statements allow you to make a decision, based upon the result of a condition.  
Ex: if,switch  
1.if  
1.1 if  
The if statement is java’s conditional branch statement. It can be used to route program execution.  
syntax:

If(condition){

Statements;  
}

Ref : Day6.Prog1  
  
1.2 if – else  
syntax:

If(condition){

Statements;  
}

else{

Statements;  
}

Ref : Day6.Prog2

1.3 Nested if:  
Ref : Day6.Prog3  


Q: WAP to print given number id even/odd?  
Q: WAP to print student results based on the grade system?  
<35 = Fail  
>=35 Pass  
35 to 65 = Third class  
65 to 80 = Second class  
80 to 90 = First class  
90 to 100 = Distension  
Q:WAP to print statements based on the dayName?  
Monday-Friday = Weekday  
Saturday-Sunday = Weendend  
   
1.4 if,else-if,else ladder:  
If(condition){  
 Statements;  
}  
else if(condition)  
{  
 Statements;  
}  
else if(condition)  
{  
 Statements;  
}  
else{

Statements;  
}

Ref day7 Prog2,3  
  
2. Switch  
Syntax:  
switch(expression)  
{  
 case value1:  
 //stmt sequence  
 break;  
 case value2:  
 //stmt sequence  
 break;  
 case value3:  
 //stmt sequence  
 break;  
 default:  
 //default stmt sequence  
}  
Nested switch  
switch(expression)  
{  
 case value1:  
 switch(expression)  
 {  
 case value1:  
 //stmt sequence  
 break;  
 case value2:  
 //stmt sequence  
 break;  
 }  
 //stmt sequence  
 break;  
 case value2:  
 //stmt sequence  
 break;  
 case value3:  
 //stmt sequence  
 break;  
 default:  
 //default stmt sequence  
}  
Ref day7 : Prog5 to 7  
  
**2. Iterative statements:**  
for, while, do-while

While  
It repeats a statement or block while its controlling expression is true.

Syntax :  
while(condition)  
{  
 //statements

}  
Q:WAP to print all the number b/w 1 to 20 in reverse order?  
Ref day8 : Prog1to 3  
 do-while:  
syntax:  
do  
{  
 //statements

}  
while(condition);  
  
Ref day8 : Prog4  
  
Variables:  
Q: What is a variable?  
A: A variable is a container which holds the value while java program is executed.  
  
Diff type of variables:  
1. Global variables  
1.1. Instance variables (Non-static Fields)  
1.2. Class variables (static Fields)   
2. Local variables  
2.1. Local variables  
2.2. Parameters  
  
There are 3 stages in variable usage:  
1. Declaration  
Syntax :   
type identifier;  
**int** studentId;  
2. Initialization  
Syntax:  
Variables name=value;  
studentId = 101;  
3. Utilization  
System.***out***.println(studentId);

1.1. Instance variables (Non-static Fields)  
Ref day8 : Prog5 and 6  
1.2 Class variables (static Fields)  
Ref day8 : Prog7  
  
Q: WAP to print area of square?  
Q: WAP to print area of circle?  
  
2.1. Local variables  
Ref day9 : Prog3  
  
2.2. Parameters  
Ref day9 : Prog4  
  
Rules for writing the variables:  
1. Variable names are case sensitive.  
2. If variable name is a single word, then it should be declared always in small letters.  
**Eg:  
int** age = 40;  
String name = "ABC";  
3. If variable name is more than a word then it should be declared in camel case.  
Eg:  
String studentName = "Student1";  
String parentName = "parent1";  
**int** roleNumber = 123;  
4. Keywords should not be used for variable names.  
**int** int = 10; ->Not allowed  
String class = "10"; ->Not allowed  
String String = "ABC"; ->allowed because string is not a keyword  
6. \_ and & symbols are allowed

**eg:  
int** parentSalary$ = 1000; allowed

**int** $parentSlary1 = 2000; allowed  
**int** \_parentSlary2 = 3000; allowed  
7. Constants are always preferred to write in CAPS.  
Eg:  
**final** **static** **double** ***PI*** = 3.142;

  
**Iterative statements:**  
for  
Syntax:  
for(Initialization;condition;iteration){  
//statements;

}  
Eg: Ref day9 : Prog5 and 6  
Q:What is a looping statement?  
Lopping statements are the statements that execute once or more statement repeatedly several number of times.  
Q: Wap to print all natural numbers b/w 1 to 10?  
Ref day9 : Prog8  
Q:Wap to print all the odd numbers b/w 1 to 10?  
1 3 5 7…  
Ref day9 : Prog9

Q:Wap to print all the even numbers b/w 1 to 10?  
2 4 6…  
Ref day9 : Prog10  
Q:Wap to print all the numbers multiple of 3 b/w 1 to 10?  
3 9  
Ref day9 : Prog11  
  
Nested for loops:  
Syntax:  
  
  
  
for(Initialization;condition;iteration){ //outer for loop  
//statements;  
 for(Initialization;condition;iteration){ //inner for loop  
 //statements;

}

}  
  
Q: WAP to print patterns? \*\*\*  
\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*  
  
Ref : day10 Prog1

Assignment  
\*\*\*\*\*  
\*\*\*\*

\*\*\*

\*\*  
\*  
  
Ref : day10 Prog2

1

23

456

78910

Ref : day10 Prog3  
A

BC

DEF

GHIJ

Ref : day10 Prog4 \*\*\*

A

AB

ABC

ABCD

Ref : day10 Prog5 \*\*\*  
A

BB

CCC

DDDD

EEEEE

Ref : day10 Prog6

0

10

010

1010

Ref : day10 Prog7

1

01

101

0101

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

Ref day11:Prog2  
  
  
0  
0 1  
0 2 4  
0 3 6 9  
0 4 8 12 16

For Each loop:

Syntax :  
for(type itr-var:collection)  
 //statement blocks  
Ref day12:Prog1  
  
Note : Nested for loops are not possible using for each loop.

3. Transfer statements:  
break, continue  
return

**Arrays**:  
Syntax:  
Declaration and Initialization:  
type[] refVar = new type[size];  
or  
type refVar[] = new type[size];

Or  
type refVar[] = {obj1,obj2….}  
**Eg : int** arr[] = {10,20,30,40,50};

Store the element into array:  
refVar[index] = value;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | variables |  |  |  |  |
| i | 10 |  |  |  |  |
|  | int |  |  |  |  |
|  |  |  |  |  |  |
| ar | 1 | 2 | 3 | 4 | 5 |

Q:What is an array ?  
A: Any array is a container object that holds a fixed number of values of a single type.  
Q:What are the different types of arrays available in java ?  
A. 1. Single dimensional array  
2. Multi-dimensional array  
2.1 Two-dimensional array  
2.2 Jagged array  
  
1. Single dimensional array  
Ref day12:Prog1  
  
Q: Wap to print sum and average of marks for a student?  
Q: WAP to print largest number in the given array? \*\*\*  
Ref day12:Prog6

Q: WAP to print smallest number in the given array? \*\*\*  
Ref day12:Prog7  
Q:WAP to check the element or number present in the array or not ?  
Ref day12:Prog8  
Q:WAP to find sum of even numbers in a given array?  
Ref day12:Prog9  
Q: WAP to sort the elements in ascending order in the given array? \*\*\*  
 Ref day12:Prog10  
Q:Wap to merge 2 arrays? //try to take 2 string arrays  
  
2. Multi-dimensional array  
2.1 Two-dimensional array

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 |
| ar | 0 | 1 | 2 | 3 |
|  | 1 | 4 | 5 | 6 |
|  | 2 | 7 | 8 | 9 |
|  | 3 | 10 | 11 | 12 |
|  |  |  |  |  |
|  |  | 0 | 1 |  |
|  | 0 | 1 | 2 |  |
|  | 1 | 4 | 5 |  |
|  | 2 | 7 | 8 |  |
|  | 3 | 10 | 11 |  |

Q:WAP to print the size of a two -dimensional array?  
Ref day13:Prog3  
2.2 Jagged array  
Ref day13:Prog6 and 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 |
| ar | 0 | 1 | 2 | 3 |  |  |
|  | 1 | 4 | 5 | 6 | 7 | 8 |
|  | 2 | 7 | 8 | 9 | 10 | 11 |
|  | 3 | 10 | 11 | 12 |  |  |

Limitations of arrays:  
1. Size is fixed. I,e once an array is created with some size it cannot be expanded I,e arrays size is not flexible.  
2. Sometimes we may waste the memory or sometimes we may need some additional memory.  
3. In arrays we can store only homogeneous data or elements  
4. There will be very less built in functionalities in arrays.  
  
4.Exception handling  
throw, try-catch, finally  
  
Q: What is an exception?  
Exception is an event that occurs during the execution of a program that disrupts the normal flow of the instructions.  
Exception are run time errors some statements will be syntactically correct but behave abnormally at the run time, then we get run time errors which are called as Exception.  
Whenever we get run time errors, program will stop its execution. In order to continue the program execution, we should handle run time errors using the **try-catch** blocks.  
  
 Whenever any java statement produces an abnormal condition at run time , jvm will create any one of the appropriate **Throwable class** type object and it will **throw** it to the method in which an abnormal statement is developed, this is called as **exception handling**  by using those try and catch blocks.

An exception can occur for different reasons, including the following:  
1. A user has entered invalid data.  
2. A file that needs to be opened cannot be found.  
3. A network connection has been lost in the middle of communication or the jvm has run out of the memory.  
  
  
**Handling run time errors using try-catch blocks is called as Exception Handling.**  
  
Risky statements should be developed inside the try block and a catch block should be written to address the exception.  
  
Hierarchy of exceptions in java:  
  
Categories of exceptions:  
1. Errors: These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in the code because you can rarely do anything about an error.  
Eg: If a stack overflow occurs, an error will arise. They are also ignored at the time of compilation of the program.  
 2. Checked exception / Compile time exception  
A checked exception is an exception that typically a user error or a problem that cannot be foreseen by the programmer.  
JVM will catch this exception and asks the programmer to fix it immediately.

|  |  |
| --- | --- |
| Exception | Description |
| classNotFoundException | class not found |
| cloneNotSupportedException | Attempt to clone an object that does not implement the cloneable interface |
| IllegalAccessException | Access to the class is denied |
| InstantiationException | Attempt to create an object of an interface or abstract class |
| InterruptedException | One thread has been interrupted by other |
| NoSuchFieldException | A requested field does not exist |
| NoSuchMethodException | A requested method does not exist |

3. Un Checked exception / Run time exception  
An unchecked exception or run time exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, run-time exceptions are ignored at the time of compilation of the program.

|  |  |
| --- | --- |
| Exception | Description |
| ArithmeticException | Arithmetic error, such as divide by zero |
| ArrayIndexOutBoudException | Array index is out of bounds |
| ArrayStoreException | Assignment to an array element of an incompatible type |
| classCastException | Invalid cast |
| IllegalArgumentException | Illegal argument used to invoke a method |
| illegalMonitorStateException | Illegal monitor operation, such as waiting on an unlocked thread |
| IllegalStateException | Environment or application is in incorrect state |
| IllegalThreadStateException | Requested operation not compatible with the current thread state |
| IndexOutOfBoundsException | Some type of index out of bounds |
| NegativeArraySizeException | Array created with negative size |
| NullPointerException | Invalid use of a null reference |
| NumberFormatException | Invalid conversion of a string to a numeric type/format |
| SecurityException | Attempt to violate security |
| StringIndexOutOfException | Attempt to index outside the bounds of the strings |
| UnSupportedOperatinException | An unsupported operation has been encountered |

Exceptions methods/functions:  
1.getMessage()   
2.getCause()  
3.toString()  
4.printStackTrace()  
  
Finally block – Finally block will be executed even if the logic under try block fails and even caught by catch block. Or even the logic presents under try block passes.  
It runs irrespective of try and catch blocks execution.  
  
Note:  
\* try block is mandatory for writing the logics of the program  
\* catch block / finally block either one is mandatory after writing the try block.  
  
throws  
throws is a keyword which Is used to re throw the already existing generated exception to the other methods.  
or  
Throws is a keyword which is used to delegate the exception to the other methods  
1.Throw keyword should be used at the method invocation  
2.Using throw keyword we can throw multiple exceptions  
3.To notify the uses of function or method about the run time error the method generation for invalid value we can use throws keyword.  
Ref to day15 Prog5  
  
Q: Give some diff b/w throw and throw’s keywords?  
\*Throws is used to notifying the JVM to handle the exception which might occur during the execution of the program  
\* throw is used to rethrow the already caught exception from the catch blocks.  
\* throw can be used to throw a new customized exception to the run time environment or JVM.  
  
Q: What is try block with resources and give some examples?  
Try block with resources:  
Any classes which are implemented by “AutoCloseable” interface then we can create those objects using try with resources blocks.  
Syntax:  
try(  
--create the objects--  
--  
)

{

--statements--  
}  
catch(Exception e){

--statements--  
}  
\* Since it is auto closeable we don’t have to use close() function to close the file.  
Eg: Ref to day16:Prog1  
  
Q: Can we have multiple catch blocks for a single try block?  
Try block with multiple catch blocks:  
Syntax:  
try{  
--statements--  
}  
catch(Exception e1){  
--statements--  
}  
catch(Exception e2){  
--statements--  
}  
Eg: Ref to day16:Prog2  
  
Q:Can we write multiple exceptions in a single catch block? \*\*\*  
We can write multiple exceptions in a single catch block by using “|” symbol  
Eg: Ref to day16:Prog3

Note: We can use “RuntimeException”, “Exception” or “Throwable” in catch block to collect the exception raised in the try block.  
Eg: Ref to day16:Prog4  
  
Q: When will be the finally block will be skipped or not executed?  
System.*exit*(0); - if we use this then main thread will be stopped or terminated so that finally block will not be executed.  
Eg: Ref to day16:Prog5  
  
  
Q: How do you write your own custom/user defined exceptions?  
Throw custom or user defined exceptions:  
Note: \*All the custom exception classes must extent to Exception class.  
\*In the catch block try to provide custom exception using throw keyword  
\*Since your saying jvm to handle this exception we should using throws keyword in the function.  
Eg: Ref to day16:Prog6 and 7  
  
OOPS   
1.Class and objects  
2.Inheritance  
3.polymorphism  
4.Encapsulation  
5.Abstraction  
  
1.Class and objects  
Q:What is a class?  
A:Class is a collection of fields and behavior.  
\*Fields are also called as Variables or attributes or Data members.  
\*Behavior is also called as Functions or methods or member functions.  
Q:What is an object?  
\*Object is an instance of a class.  
  
class syntax:  
class className{

type instance = variable1;  
 type instance = variable2;  
 ----  
 type instance = variableN;  
  
 type methodName(parameters)  
 {  
 --body of the method

}  
 type methodNameN(parameters)  
 {  
 --body of the method

}

}  
Note: The data or variables defined within a class are called as Instance variables.  
**Members of a class**:  
The methods and variables defined within a class are called members of a class.  
There are 2 types of members:  
1. Static member  
2. Non-static member  
  
1. Static member  
\* Any members declared with the keyword static is called as static member.  
\* static members include static variables, static methods and static initialization (SIB)  
\* To access the static members of a class we should use className.staticMemberName, because static members will be loaded onto static pool by class loader and the static pool name would be similar to class Name.  
  
2. Non-static member  
\* Any members declared without the keyword static is called as non-static member.  
\* Non-static members includes non-static variables, non-static methods and non-static blocks or instance initialization block (IIB)  
\* To access the non-static members of a class we should “create the object” of the class.  
\* To access the non-static members of a class we should use  
 Class-name refVariable = new Class-name();  
refVariable.non-static member  
  
Note:  
\* new operator is used to create new address space or non-static pool inside the heap memory  
\* Class-name() will load all the non-static members of a class onto the address space created by the new operator.  
\* Reference variables will store the address of the object so that we can refer of to the object.

  
Ref:day17:Prog1 and Prog2  
Q: Wap to perform swapping of 2 numbers stored in the variables using temp variable.  
Ref:day17:Prog3  
Q: Wap to perform swapping of 2 numbers stored in the variables without using temp variable.  
Ref:day17:Prog4  
Note :   
\* Global variables will reside in the heap memory/static memory, it will be there until the end of the program.  
\* If its static variable, it will reside till the end of the program or non-static means it will be there in the object space.  
   
**Functions or methods or member functions**:  
Q:What is a function in java?  
A:A java function is a collection of statements that are grouped together to perform an operation and executed whenever called or invoked.  
Syntax:  
Access-level modifier return-type method-name(arguments)

{  
----Body of the method---

}  
Access-level – public, protected, default, private  
Modifier – static or non-static  
Return-type – primitive data type, non-primitive data types  
method-name – is an identifier can be any valid identifier  
arguments – inputs for the methods



Q: WAP with given 2 integers or numbers return true if sum of them is 30 or one of them is 30.  
Ref day 18 : Prog3  
Q: WAP with given 2 numbers return twice there sum if both are same otherwise return their sum.  
Ref day 18 : Prog4  
Q: There are 2 monkeys, if both monkeys are smiling then we are in trouble, if both the monkeys are not smiling then also we are in trouble, return true if we are in trouble.  
Ref day 18 : Prog5  
  
Constructor:  
Constructor is a special function/method which is used to construct a object or create a object for a class and cannot return or does not have return type.  
Whenever a object or a class is created using new keyword then the constructor without arguments will be created by default.  
  
Q: Why constructor?  
A: We can use constructor to create object, without constructor we cannot create a object.  
  
Rules for writing a constructor:  
1. Constructor name should be same as class name.  
2. Constructor should not have return type.  
3. Constructor is also called as non-static initializer.  
4. Constructor should be there in every class.  
5. If the programmer dint add constructor then jvm will add default constructor.  
  
Q: Give some diff b/w method and a constructor?

|  |  |
| --- | --- |
| Method | Constructor |
| Method can have any name | Constructor name should be same as class name |
| methods can have return type | Constructor should not have any return type |
| methods can have static or non static | Constructor should be always non static |
| methods may or may not return values | Constructor should not return any value |
| java does not provide any default method | java provides default constructor |
| methods cannot be used for object creation | Constructor will be used for object creation using new operator |

Constructor over loading – is used for initializing the values.  
Q: Can we have multiple constructors inside a single class?  
A: Yes but at least arguments should be different.  
  
**2.Inheritance**  
What is inheritance?  
\* Getting the features or properties of one class from another class is called as Inheritance.  
\* Inheritance is a mechanism in which one class acquires all the properties and behaviors of another class with the specific relationship.  
\* The class from which other class acquires the features is class as Super or Parent class.  
\* The class which acquirees the features or properties is called as Sub class or Child class or derived class.  
  
What are the advantages of Inheritance?  
\* Extensibility  
\* Code optimization  
\* Code reusability  
\* Code maintainability  
  
How do you achieve inheritance?  
\* We can achieve inheritance by using “extends” keyword.  
  
How many types of inheritance present in java?  
1. Single level inheritance – one class inheriting from only one super class  
2. Multi-level inheritance – One class inheriting from another sub class  
3. Hybrid inheritance – Two or more class inheriting from common super class   
4. Multiple inheritance – One class inheriting from multiple immediate super class  
  
Note: Multiple inheritance is not possible in java through classes instead this can be achieved via Interfaces.  
  
1. Single level inheritance  
 

2. Multi-level inheritance



1. Multiple inheritance  
   
2. Hybrid inheritance  
     
   super keyword  
   Super keyword is used to access super class non-static members in case of inheritance between the classes.   
     
   Super calling statement:  
   \* super calling statement is used to call super class constructor in case of inheritance is being established.  
    \* super calling statement will be immediate super constructor in case of inheritance  
   \* Through super calling statement we can achieve constructor chaining  
   \* Constructor chaining means sub class constructor calling immediate super class constructor.  
   Ref : Day 20 : A, B ,C  
    

Super.read() – parent class function

**IIB & SIB**IIB –Instance initialization block  
  
What is IIB and explain their uses?  
\*IIB are executed when objects are created, the numbers of times we create the objects same numbers of times IIB will be called/executed.  
\*IIB are used to initialize all the instance variable in one place and give us better readability of code.  
\*IIB purpose is to initialize all non-static members in one place for readability.  
Note: \***IIB will be first called before the execution of the constructor**.  
\*IIB will be first called because initialize variable then load the object into the heap memory.  
Syntax:  
{

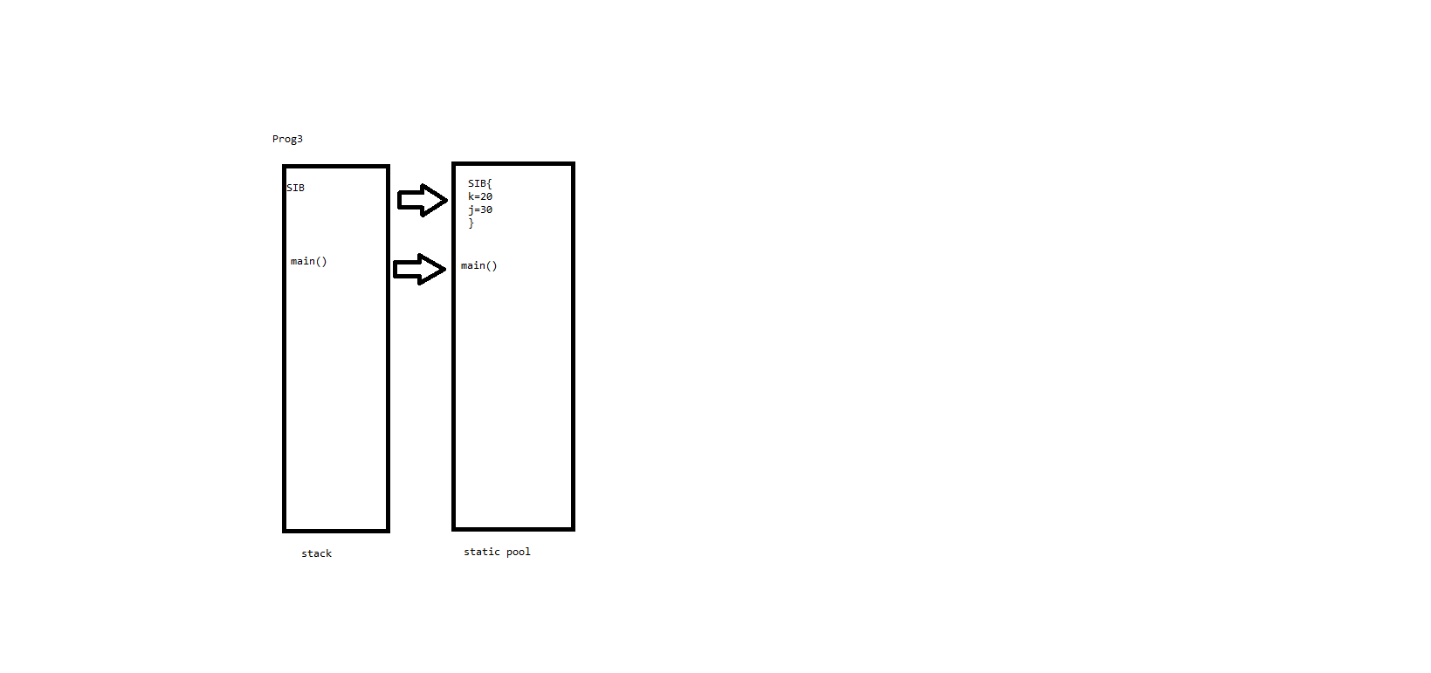
//-----steps/initialization----

}

Eg: Ref:day21:Prog1  
  
Note: We can initialize both static and non-static variable inside the IIB.  
  
Eg: Ref:day21:Prog2  
  
SIB – Static initialization block  
What is SIB and explain their uses?  
\* Static initialization block runs first before the main() function/method is called.  
\* SIB does not require any invoking statement and run only once.  
\* In SIB instance variable initialization in not possible.  
\* We cannot initialize non-static variables inside the SIB.  
  
Main diff b/w SIB and IIB?   
\* IIB can initialize both static and non-static variables  
\* SIB can initialize only static variables  
\* IIB will be called as many number of times we create the object  
\* SIB will be executed only once before main() method is called.  
  
Q:Can we write a program without using PSVM()?  
Yes – SIB  
  
syntax:  
**static** {

--initialization--

}

  
Note: this keyword is used to differentiate between local and global variables.

|  |  |
| --- | --- |
| this calling statement | super calling statement |
| is used to call current class constructor | is used to call the super class constructor |
| inheritance is not mandatory | will be used only when we have inheritance |
| will no default this calling statement added by the compiler | compiler develops a default super calling statement |
| Through this calling statement we can utilize initializing code / date members to another constructor | Through super calling statement we can achieve constructor chaining |
| this calling statement is not mandatory | super calling statement is mandatory in case of inheritance |

Interview questions:  
  
What is JAVA?

What are the advantages of java?

Explain Java Architecture?

What are the advantages of package?

What is import section in java?

What is class?

What are the different members of class?

Explain few rules while creating identifiers?

What is a keyword in java?

What is a data type in java?

What is implicit and explicit casting?

What is an array?

What is jagged array?

What is an exception?

Can we write a program without a catch block?

Explain the hierarchy of exceptions?

Give some examples for unchecked exceptions?

What is a finally block in java?

Give some diff b/w throw and throw’s keywords?

How do you write your own custom/user defined exceptions?

What is Inheritance?

Give some diff b/w method and a constructor?

Explain why Multiple inheritance is not possible in java?

Give some diff b/w this and super keyword?

Give some diff b/w SIB and IIB?

Programs

1.WAP to print given number is even/odd?

2.WAP to print all the numbers b/w 1 to 10 in reverse order?

3.WAP to print below pattern?

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

4. WAP to swap 2 numbers without using third variable?

5. WAP to find factorial of a given number?

6. WAP to find common elements b/w two arrays?

7. WAP to remove the duplicates from an array?

8. WAP to find largest and smallest element in an array?

3.polymorphism – poly(many) morphism(forms)  
Q: What is polymorphism?  
Polymorphism is the ability of an object to take one or many forms.  
Q: What are the different types of polymorphism present in java?  
There are 2 types of polymorphism present in java:  
1. Compile time polymorphism  
2. Run time polymorphism  
  
1. Compile time polymorphism  
\*Compile time polymorphism is a type of polymorphism which can be achieved at java compilation time.  
\*It is also referred as **static polymorphism or early binding**.  
\* Compile time polymorphism can be achieved throw **method overloading** in java.

**method overloading**Q: What is method overloading?Definition:  
1. Developing multiple methods with the same name, but variations in the arguments list.  
1. Number of arguments  
2. Types of arguments  
3. Piston/order of arguments  
  
When ever we need to perform the common operation or task with the variations in the inputs, we should go for method overloading.  
  
Q: What are the advantages of method overloading?  
Through method overloading:  
1. We can achieve consistency in the method names, which are developed for common purpose.  
2. It is easy to remember method names.  
3. We can achieve efficiency in the program reading.  
4. We can achieve compile time polymorphism through overloading.  
5. While overloading method names should be some variations in arguments list, other parts of method does not matter. i.e return type, modifiers, access level can be different.  
6. We can overload static and no-static methods.  
  
Eg: println() is the function which accepts multiple arguments and this the best example for method overloading.  
\*\*\*  
Q: Can we overload the main method?  
Yes, we can but the program execution with main method having arguments as String args[], other versions of the main method should be invoked by the programmer explicitly.  
Ref to day22:Prog1,2,3.  
  
2. Run time polymorphism  
\* Run time polymorphism is a type of polymorphism which can be achieved at java run time or execution time.  
\* It is also referred as dynamic polymorphism or dynamic method dispatch.  
\* Run time polymorphism can be achieved thorough **method overriding** in java.   
**method overriding**Q: What is method overriding?  
Changing the implementation of super class method in the sub class according to the needs of the sub class in called as method overriding.  
  
To achieve method overriding 3 types are mandatory:  
1. Inheritance.  
2. Non-static methods  
3. Signature or method declaration should be same.  
  
Whenever we perform method overriding super class implementation of the method will be lost or masked in the sub class. i.e,, we will get the latest implementation of the method.  
  
Q: What are the advantages of method overriding?  
1. We can improve performance.  
2. We can achieve, standardization, run-time polymorphism.

Ref to day22:Prog 4,5.  
  
Note: By using super keyword we can execute both parent class function and child class function as well.  
 Ref to day22:Prog 6,7,8,9  
  
Q: What are diff b/w method overloading and overriding? \*\*\*

|  |  |
| --- | --- |
| method overloading | method overriding |
| Overloading can be achieved with and without inheritance. i.e. inheritance is not mandatory | Inheritance is mandatory |
| Overloading can be achieved on both static and non - static methods | We can override only non-static methods |
| For overloading, method name should be same with variations in arguments. | For overriding complete method declaration should be same including arguments. |
| Through overloading we can achieve compile time polymorphism | Through overriding we can achieve run time polymorphism |
| we can overload main method | we cannot override main method |
| We can achieve constructor overloading | we cannot override constructor |

**Encapsulation**Q:What is Encapsulation?Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.  
Encapsulation describes the ability of an object to hide the data and methods from the rest of the world.  
  
Restriction the direct access to the data members and giving indirect access to the data members through the getter and setter methods is called as Encapsulation.  
  
While performing encapsulation the data members will be declared as private and getter and setter methods will be declared as public.  
  
**Getter are public method which will give the data and setter are public method which will manipulate the data.**Conceptually they are getter and setters, but method name need to be always get and set it’s a standard industry convention to name gets and sets with get and set followed by private data member name.  
  
Getters are also called as **Accessors**.  
Setters are also called as **Mutators**.  
Q: What are the advantages of using Encapsulation?  
Through encapsulation we can achieve data security. i.e. we can have complete control over the data which will be manipulating.  
Here the data and code which will be manipulating the data will be binded in the same class.

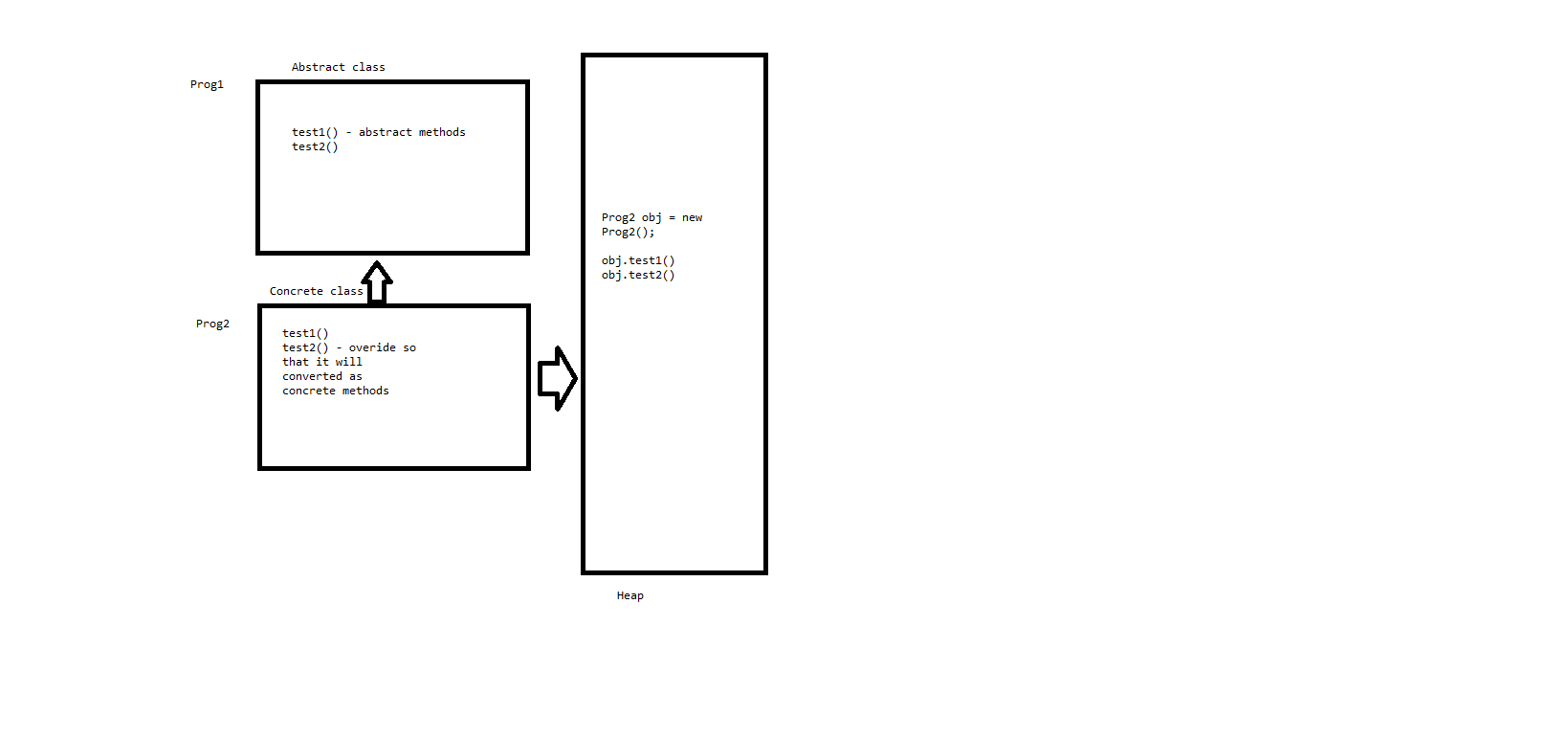
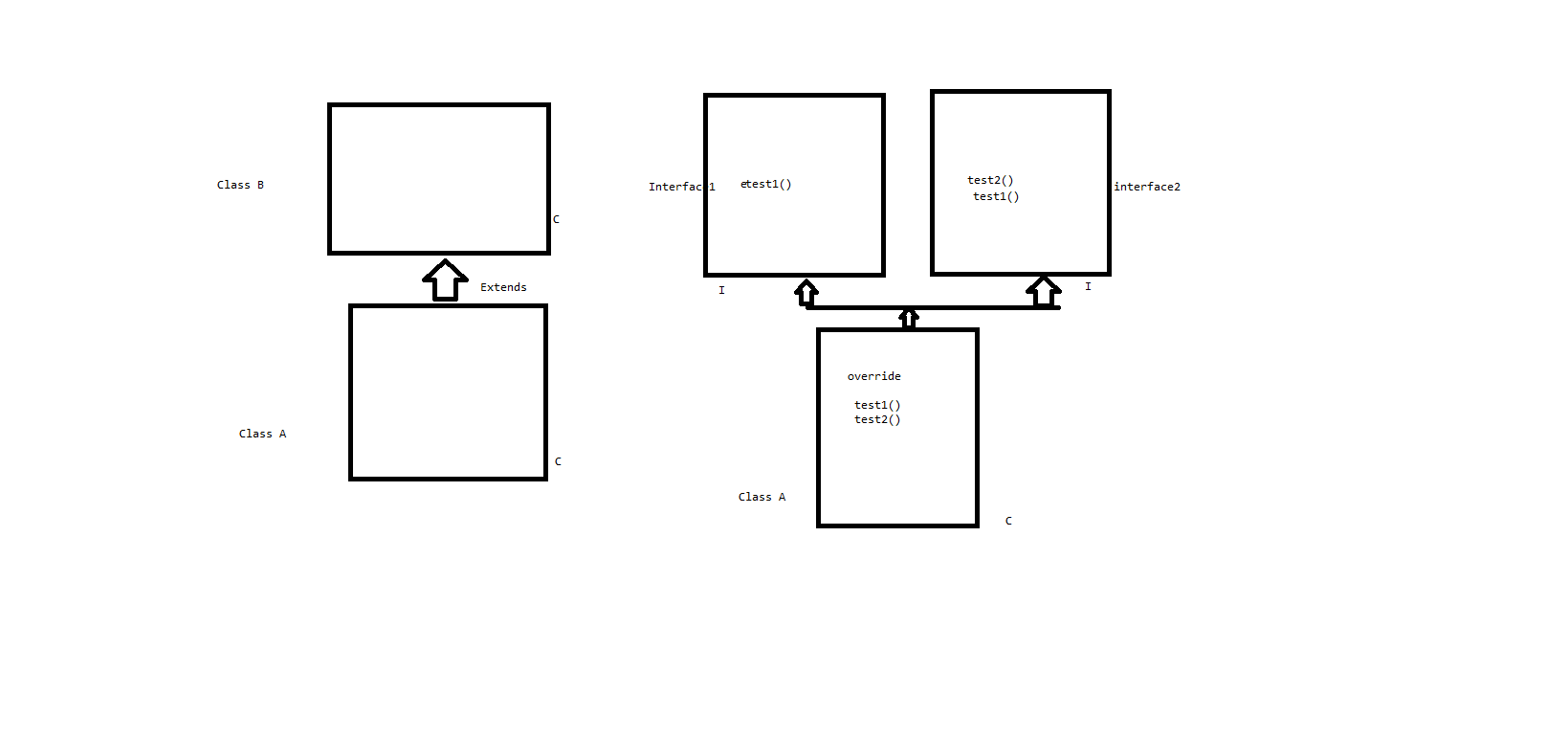


**Java Bean class:** Any class which is having getter and setter methods is called as Java Bean class.  
Developing a class with private data members and giving access to those private data members using public service methods getters and setters is called as Java Bean class.  
  
Steps to generate getters and setters:  
1. Declare the private data member.  
2. Right click on the java class and click on source and click on Generate getters and setters.  
3. Select the getter and setter methods and click on finish.  
Ref to day23:Prog 1,2,3  
  
**Access Levels / Access modifiers:**  
Q: What is access modifier?  
Access modifiers defines the scope/visibility of the members of a class (variables, constructors and methods)  
Access modifiers are used to put restrictions on the members and through access levels we can achieve encapsulation.  
  
Q: What are the list of access levels available in java?  
There are 4 levels:  
1. Public  
2. Protected  
3. Default  
4. Private  
  
1. Public: Public members will have application-level access i.e., public members can be accessed from any program of any package.  
Note: public is most visible access levels.  
2. Protected: Protected members can be accessed within the package and can be accessed outside the package through inheritance.  
  
3. Default: Default members will have package level access i.e, default members can be accessed only within the current package.   
Note: If any members are not declared with any other access level, then that data members will have default access levels. There is no keyword called default.   
4. Private: Private members will have class level access i.e private members can be accessed only within the class it is created.

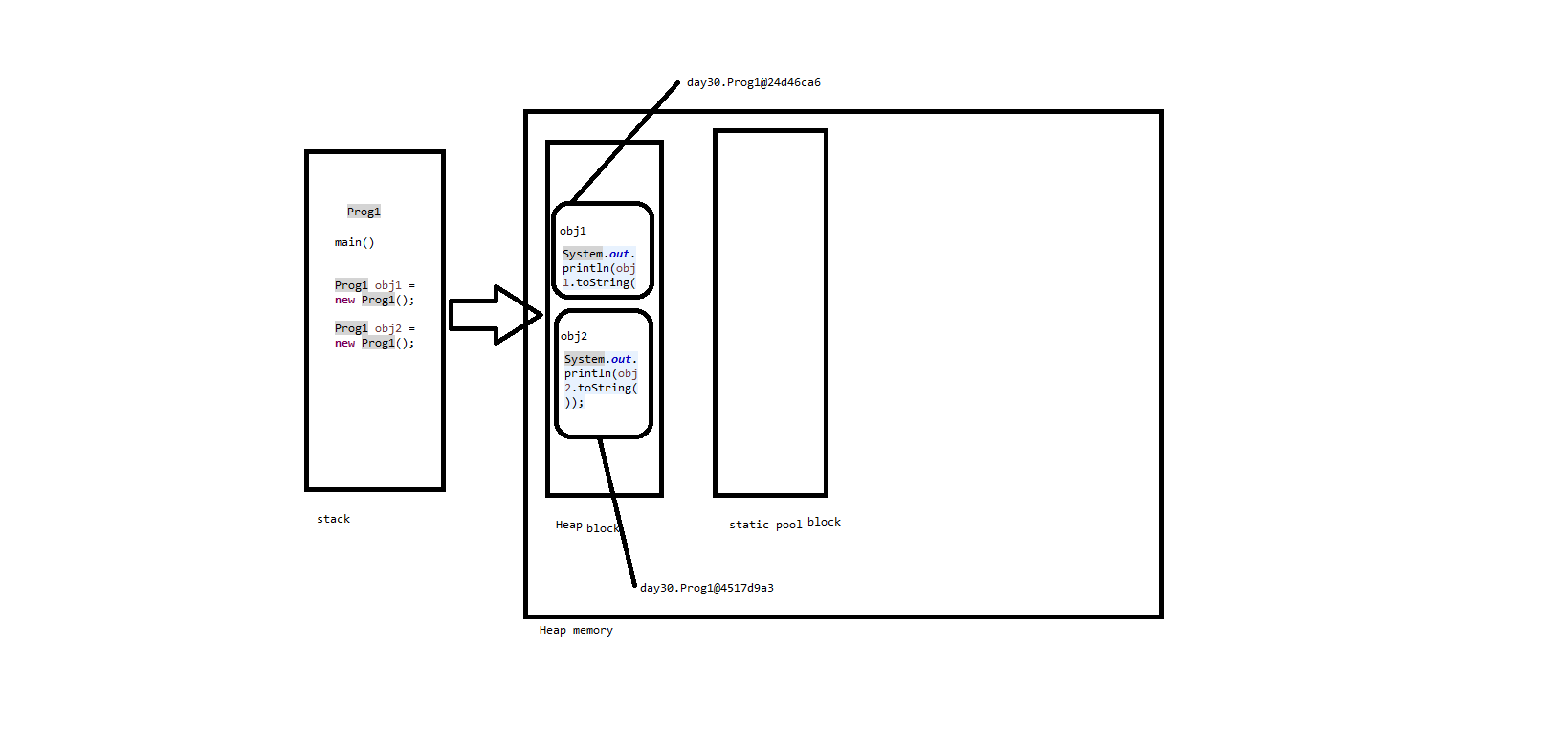
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modifier | Class | Inside Package | Outside Package | Subclass | World |
| Public | Y | Y | Y | Y | Y |
| Protected | Y | Y | N | Y | N |
| default | Y | Y | N | N | N |
| private | Y | N | N | N | N |

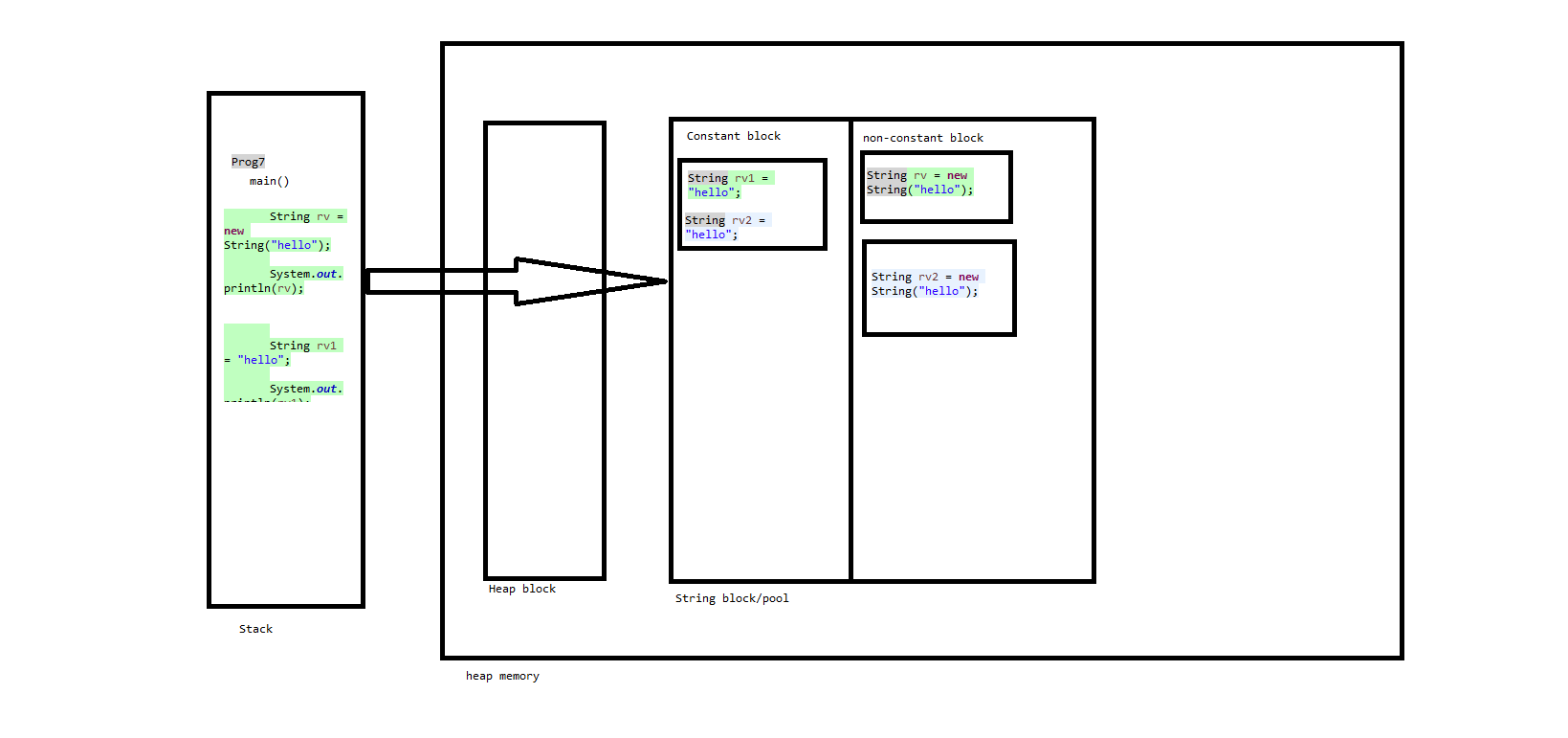
Note:   
\* A class cannot be declared with private and protected access levels.  
\* A class can have only 2 access levels public and default.  
\* Private and protected access levels are only for members of a class not to a class itself.  
\* Public classes can be imported through import statement but default classes cannot be imported.  
  
**Abstraction - Abstract class**Concrete methods:The method which will have declaration and definition together is called as Concrete methods.  
Eg:   
void test(){

----stmt’s----  
}  
Concrete class: The class in which we can develop concrete methods is called as concrete class.  
Eg:  
class A  
{  
void test(){

----stmt’s----  
}  
}  
Abstract methods: The methods which are just having declaration without definition in class is called as Abstract methods.  
\*These abstract methods will use abstract keyword.  
Eg:  
abstract void test();  
Abstract class: The class in which we have an option to develop abstract methods is called as abstract class.  
\*These abstract class will use abstract keyword.  
Eg: abstract class A{  
abstract void test();  
}  
  
Q: What is abstraction?  
Abstraction is a process of hiding the implementation details and showing only the functionality to the user.  
  
Note:  
\* We cannot create object to the abstract class or in other words abstract class cannot be instantiated.  
\* Abstract class should always have a sub class and the sub class should give implementation or definition of the inherited abstract class.   
\* Giving implementation or a new definition to the inherited abstract class in the sub class is also called as method overriding.  
  
Q: What are the advantages of using abstract class?  
 1. Standardization  
2. Abstraction  
  
\*\*\*  
Note: \*Abstract class can have both abstract methods as well as concrete methods.  
\* On inheritance we will have 2 options:  
1. Make the sub class also as abstract class  
2. Or override all the abstract methods in the sub class.  
\* Once override of abstract methods will happen these methods will be converted as concrete methods so that we can create the object of sub class.  
\* Abstract classes can have constructors.  
\* Abstract class is not 100% abstract because it can have concrete methods as well.  
  
  
  
  
  
Q: What are the rules to be followed while implementing abstract classes?  
Rule 1: The sub class should override all the inherited abstract methods.  
Rule 2: If the sub class does not override all the abstract methods then sub class should be declared as abstract.  
  
Q: Can we declare an abstract method as static?  
No  
Q: Can we declare a abstract method as final?  
No  
Q: Can abstract class as static / final?  
No  
Q: Will abstract class will have constructor?  
Yes  
Q: What are the advantages of using abstraction?  
1. Group all the common features in the abstract class.  
2. Override all the common features based on some variations in the sub classes.  
3. Create the object of sub class and call the features/methods accordingly.  
  
Java Types:  
There are 4 types of java types available in java:  
1. Class  
2. Interface  
3. Annotation  
4. Enumeration  
Annotation  
Q: What is annotation?  
Instructions given by the programmer to the compiler specifying what a compiler should do during compilation.  
\*They are used to give implementation to the developer, compiler and run time environment.  
  
Q: How many types of annotations we have in java?  
1. @Override  
2. @Supresswarnings  
3. @Deprecated  
  
1. @Override – Helps us to check whether we are overriding the method or not.  
Ref Day 27 Prog1 and 2  
2. @Supresswarnings – Helps us to suppress warning messages in the program.  
Ref Day 27 Prog3  
3. @Deprecated – Helps us to notify that a particular method is not in use or going to be decommissioned in future releases.  
Ref Day 27 Prog4  
  
2. Interface  
Q What is an interface?  
An interface is a blue print of a class  
\* Interface is java type which is 100% abstract. Through interface we can achieve multiple inheritance in java up to some extent.  
\* Through interface we can achieve standardization and abstraction.  
\* We cannot create object of an interface.  
  
\* If a class inheriting from interface, then we should user “implements” keyword  
\* In interface all the methods will be automatically public  
\* In Interface all the methods will be automatically abstract i.e. we don’t have to use abstract keywords.  
\* Interface will not have a constructor so that we cannot create an object for the interface.  
\* **A class can implement multiple interfaces.  
\* An Interface can extends multiple interfaces by using extends keyword.**  
  
Note:  
class to class – extends  
class to interface – implements  
interface to interface – extends  
  
Note : We can have static functions, private static functions and default functions inside the interface in latest versions of java i.e., java >18  
  
\*\*\*  
\* **Interface can have abstract, static, default and private static methods in latest versions of java.**Ref : day 28 TV, Samsung, Sony example  
Q: Can a interface inherit from another interface ?  
YES by using extends keyword.  
Q: Can one interface inherit from multiple interface?  
YES  
Q: Can one interface inherit from concrete class or abstract class?  
NO, because interface is 100% abstract and in this case it might inherit concrete methods.  
Q: List some difference b/w abstract class and interface? \*\*\*

|  |  |
| --- | --- |
| Abstract class | Interface |
| Abstract class in not 100% abstract | Interface is 100% abstract |
| In abstract class we can have constructor | We will not have constructor |
| Through abstract class we cannot achieve multiple inheritance | Through interface we can achieve multiple inheritance |
| In abstract class it Is mandatory to declare abstract methods with abstract keywork | In Inheritance it is not mandatory |
| In abstract class methods will have programmer level access | In interface by default all the methods will be public |
| If a class is inheriting from abstract class constructor chaining is required | If a class is inheriting from an interface constructor chaining is not required |
| In abstract class we can have concrete methods | In interface abstract, static, default and private static methods are allowed |

4. Enumeration (enum)  
Q: What is an enum?  
They are used to group the fix number of constants.  
Q: What is the use of enum?  
Through enum we can achieve uniformity.  
  
Syntax:  
enum enumName  
{  
---constants---  
}  
  
enum months  
{  
---Jan  
--Feb  
--  
--  
--Dec  
}  
Ref day 29: Prog1,2  
  
**Object class  
\*** Object is the built in class present in java.lang package  
\* Object class is the super most class for all the classes present in the java.  
\* Any class is the sub class of object class.  
\* Object class contains only one default constructor with no arguments.  
\* There are lot of non-static methods in object class which will be part of every class in the java.  
\* We don’t have to import object class for all the classes in java, java compiler will takes care of it.  
  
Object class has few non-static methods :  
**1. toString()  
2. Equals()  
3. hashCode()**  
4. Notify()  
5. notifyAll()  
6. Wait()  
7. getClass()  
8. Clone()  
9. **Finalize()**  
  
**1. toString()**\* toString() is a non-static method of object class which will be inherited to every class in java.  
\* toString() will be available to every object of java.  
\* toString() will give the object information in the string format i.e, it will give the string representation of the object on which it is called.  
\* String representation will contain fully qualified class name with converted hexadecimal numbers for the memory address of object with a separator @.  
\* Fully qualified class name means packagename.class name  
\* whenever we print any reference variable in java there will be a internal to the toString()   
\* toString() will always be over ridden to give the detail information of object including attributes and its values.  
Eg: day29.Prog3@24d46ca6  
Ref day 29: Prog3  
Ref day 30: Prog1,2  
  
  
  
**2. Equals()**\* equals() is a non-static method of object class which will be inherited to every class of java.  
\* equals() will be available to object of java.  
\* equals() will compare 2 objects equality based on memory address. If memory address are same, it returns true otherwise false.  
\* Any sub class can override equals()  
\* equals() will always be overridden to compare 2 objects equality based on attributes.  
  
Q: What is the diff b/w == and equals()?  
== is used for comparing 2 values  
equals() is used for comparing memory address of 2 objects.  
Ref day 30: Prog3,4  
  
**3. hashCode()**\* hashCode () is a non-static method of object class which will be inherited to every class of java.  
\* hashCode () will be available to object of java.  
\* hashCode() will give unique integer number based on the memory address of the object i.e., it will give the integer representation of the object.  
\* The unique integer number will be generated by using some hashing algorithm.  
\* The unique number is also called as Hash Code.  
\* Always hashCode() will be overridden to generate the hash number by using the attributes of the object.  
 Ref day 30: Prog5,6  
  
**Strings class**  
\* Strings is a concrete class in java.  
\* It is also a built in class in java.  
\* String class is developed to handle string literals and string values.  
\* To use string class in any other package there is no need to import string class explicitly.  
  
There are 2 ways to represent the strings in java:  
1. String rv = new String(“Hello”);  
2. String rv = “Hello”;  
  
Note:  
\* String variables are reference variables.  
\* String is class and even it is a sub class to object class.  
\* All the non-static methods of the object class will be inherited to string class.  
\* The 3 methods toString(), hashCode(), equals() functions will be overridden in string class.

String pool:  
1. All the string objects will be stored in string pool of the heap memory.  
2. String pool is divided into 2 parts:  
2.1 Constant pool:  
\* All the string objects created using “=” will be stored in constant pool.  
\* Duplicates of 2 string objects are stored in the same memory blocks.  
2.2 Non-constant pool.  
\* String objects which are created using new operator will be stored in non-constant pool.  
\* In non-constant pool duplicates are allowed.  


Q: String is immutable explain?  
\* String is immutable means their values cannot be changed once they are created.  
\* If we try to change the string java does not modify the original one. Instead, it creates a new string object and stores it in Non constant block.

. Ref day 31: Prog1-4  
  
String functions :  
length() -> return the int with number of characters in the string. Even space is considered as a character in the string.  
charAt(index) ->   
\* Always the index will starts with 0 and if we are trying to access the index which is not there then we can expect java.lang.StringIndexOutOfBoundsException.  
indexOf(char) -> \* returns the int i.e, index of the first matching character in the string.  
toLowerCase() and toUpperCase() -> \* Converts all the characters in the string into lower or upper cases.  
startsWith(“String”) -> returns true if the String is matching to the first part of actual string otherwise it will returns false.   
endsWith(“String”) -> returns true if the String is matching to the last part of actual string otherwise it will returns false.  
contains(“String”) -> returns true if the expected string matches to the part of the actual string otherwise it will returns false.  
substring(index) -> returns String from provided index till the end of string all the characters will be printed.   
substring(start Index, end Index) -> returns String from provided start index till the end Index all the characters will be printed.   
  
\*\*\*  
isEmpty()-> returns true if the actual string in empty otherwise it returns false here space is also considered as a character.  
\*\*\*  
isBlank ()-> returns true if the actual string does not contain characters [A-Z,a-z,1-9 or spl characters] otherwise it returns false here space is not considered as a character.

equalsIgnoreCase(String)-> returns true if expected text Matches with the exact actual text with ignoring the cases i.e, it is not case sensitive.

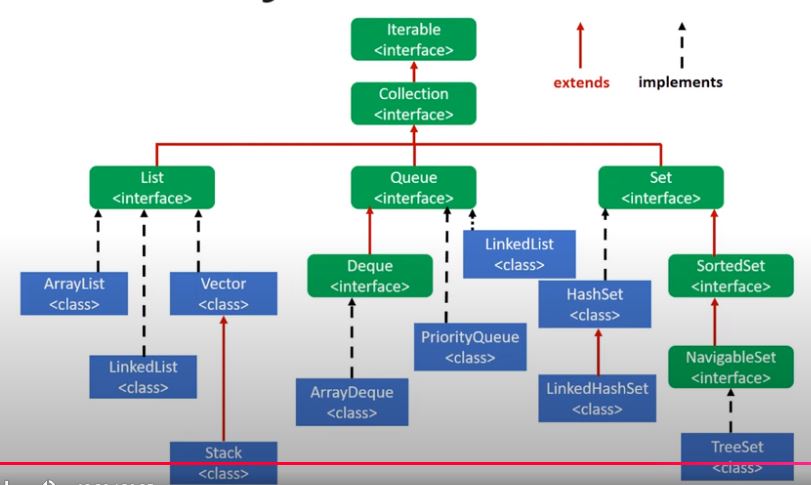
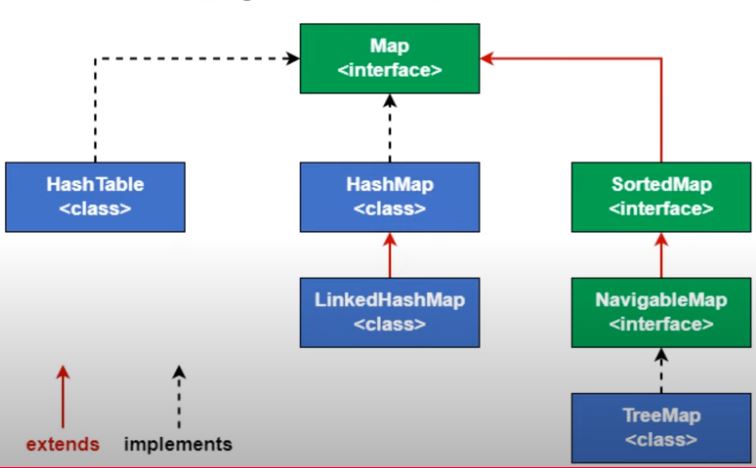
equals() )-> returns true if expected text Matches with the exact actual text without ignoring the cases i.e, it is case sensitive.  
  
trim() -> used to remove the white spaces (pre or post spaces) present in the string.  
  
replace(‘old character’, 'new character') -> replace old character with new character in the given string.  
  
toCharArray() -> used for converting the string into an array type of characters.  
  
split(string/reg expression) -> return type of split function is array of strings, which will be obtained based on the parameters (string/reg expression).  
Ref day 32: Prog 4 and 5  
  
Integer.*valueOf*(String) -> used for converting string to integer data type.  
  
String.*valueOf*(data type) -> used for converting int/float/double/char/boolean.. to String data type.  
  
Arrays.*toString*(array obj) -> used to convert array object into string type.  
  
Ref day 31: Prog 5  
Q: WAP to print the string in reverse order?  
Ref day 31: Prog 6  
Q: WAP to check given string is palindrome or not?  
Ref day 31: Prog 7  
Q: WAP to print occurrence of char “A” in the given string?  
Ref day 32: Prog 1  
Q: WAP to replace char “A” with “G” in the given string?  
Ref day 32: Prog 3  
Q: WAP to count vowels and consonants in the given string?  
Ref day 32: Prog 9  
Q: WAP to print unique characters in the given string?  
Ref day 32: Prog 10  
Q: WAP to remove space from a given string?  
Ref day 33: Prog 3  
Q: WAP to print each letter twice from a given string?  
Ref day 33: Prog 4  
Q: WAP to give 2 o/p “abcde” and “ABCDE”  
given input is “aBACbcEDed”  
  
String – immutable (values cannot be changed)  
StringBuffer and StringBuilder – mutable (values can be changed)  
  
StringBuffer and StringBuilder holds the same type of functions in java  
1. compareTo – returns 0 if all the characters in obj1 is similar to obj2.  
2. append(“String”) – used to append the string at the last of the original string.  
3. Insert(index,”String”) – appends the string based on the provided index.  
4. delete(start index, end index) – removes the string based on the provided index.  
5. reverse() – reverse all the characters present in the string.  
6. capacity() – print the default capacity of the memory block.  
7. ensureCapacity(int) – increase the capacity of the required memory block.  
  
Note: \* Each object you create using string buffer and string builder a new memory block will be allocated even the provided string is same.  
\* Each time value or data modified will replace the data or value of the original memory block.

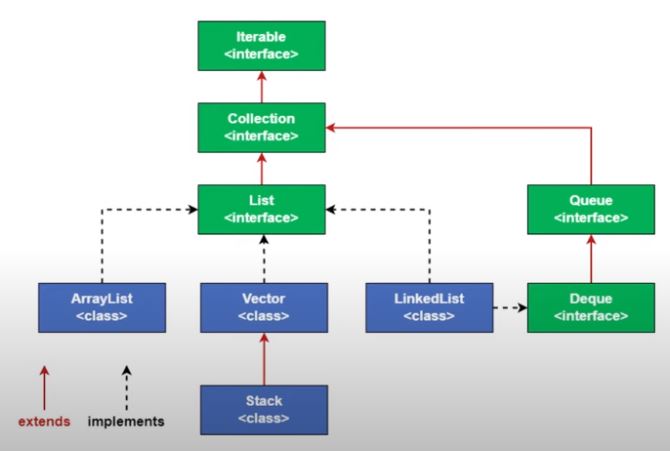
|  |  |  |  |
| --- | --- | --- | --- |
|  | String | String buffer | String builder |
| Java version | java v1.0 | java v1.0 | java v1.5 |
| Storage area | Heap - constant and non-constant pool | Heap - non constant block | Heap - non constant block |
| Mutuality | Immutable | Mutable | Mutable |
| Thread-safe | Yes | Yes | No |
| Synchronized | No | Yes | No |
| Performance | Slower | Faster | More faster |
| Capacity | NA | 21 | 21 |
| Best suite for | Small modifications | more modifications | more modifications |
| Additional operations | Concat, equals& equals ignorecase, split | reverse, append, insert, delete and comparison | reverse, append, insert, delete and comparison |

Object casting:  
Casting the object from one class to another class type or interface is called object casting.  
  
\* To achieve object casting inheritance is mandatory. (is-a)  
There are 2 types of castings:  
1. Up casting  
2. Down casting  
  
1. Upcasting: Converting the sub-class object to super class type or interface is called as upcasting.  
Whenever we perform upcasting sub class features will be hidden and we can access only super class features.  
Note: If it’s a case of overriding then child class latest method will be executed.   
  
2.Downcasting: Converting super class object to sub class type is called as down casting.  
\* Whenever we perform upcasting sub class features will be hidden.  
\* So to access these hidden features we perform down casting.  
\* To achieve this down casting inheritance is mandatory.  
\* Direct down casting is not possible we should perform upcasting first and then only we can perform down casting.

\*\*\*\*\*\*  
Collections:  
  
\* Collections is a framework or API, in which lot of classes, interfaces and build in functionality which will serve common purpose.  
  
\* All the collections framework functionality has been implemented using the standard data structures like stack, queue, single and double linked list.  
  
\*All the collection framework related classes and interface are available in java.util package.  
  
\* In collection framework size will be dynamic and we can store heterogenous data/elements.  
  
\* To overcome the disadvantages of arrays, collections is introduced.   
  
\*Collections is also called as “Dynamic array”.  
  
\*\*\*  
Q: What is difference b/w collection and collections in java?  
\* **Collection** is a root **interface** in collections framework extended to Iterable interface.  
  
\* **Collections** is a concrete class /wrapper class in collections framework which provide some utility methods like sorting, searching etc..  
  
\* Based on the requirements collections is divided into 4 parts:  
1. List  
2. Queue  
3. Set  
4. Map  
  
Sub categories:   
1. List  
ArrayList  
LinkedList  
Stack  
Vector

2. Queue  
Deque  
ArrayDeque  
PriorityQueue  
  
3. Set  
HashSet  
Linked HashSet  
SortedSet  
NavigableSet  
TreeSet  
  
4. Map  
HashTable  
HahMap  
LinkedHashMap  
SortedMap  
NavigableMap  
TreeMap

Architecture of collections framework:  
  
contd..  
  
  


**List**

**Vector**  
Major operations of vector collections are:  
1. Creation of vector  
2. Addition of elements into the vector  
3. Retrieval of elements from the vector  
4. Deletion of elements from the vector  
5. Verification of elements from the vector  
6. Updating of elements in the vector  
  
   
\* default capacity of vector is 10  
\* default size of the vector is 0  
\* If we add more than 10 members into the vector then vector will create the new array with exact double in size and copies all the objects from the existing array to new array and destroys the old array.  
\* After removing an object from the vector all the other objects will be shift lifted in order to make the clean array and size of the array will not be decreased.   
\* Duplicates are allowed in the vector  
\* If you add homogeneous data types then try to add generics to the vector.  
\* If you want the default capacity of 30 then overload the vector constructor with capacity.  
Vector<String> v1 = **new** Vector<>(30);  
\* Special functions available are firstElement(), lastElement(), capacity()

0 1 2 3 4 5 6 7 8 9

vector v1 1 2 3 4 5 6 7 8 9 10

destory

v1

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20   
  
**ArrayList**Major operations of ArrayList collections are:  
1. Creation of ArrayList  
2. Addition of elements into the ArrayList  
3. Retrieval of elements from the ArrayList  
4. Deletion of elements from the ArrayList  
5. Verification of elements from the ArrayList  
6. Updating of elements in the ArrayList  
  
ArrayList al1 0 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 10

destory

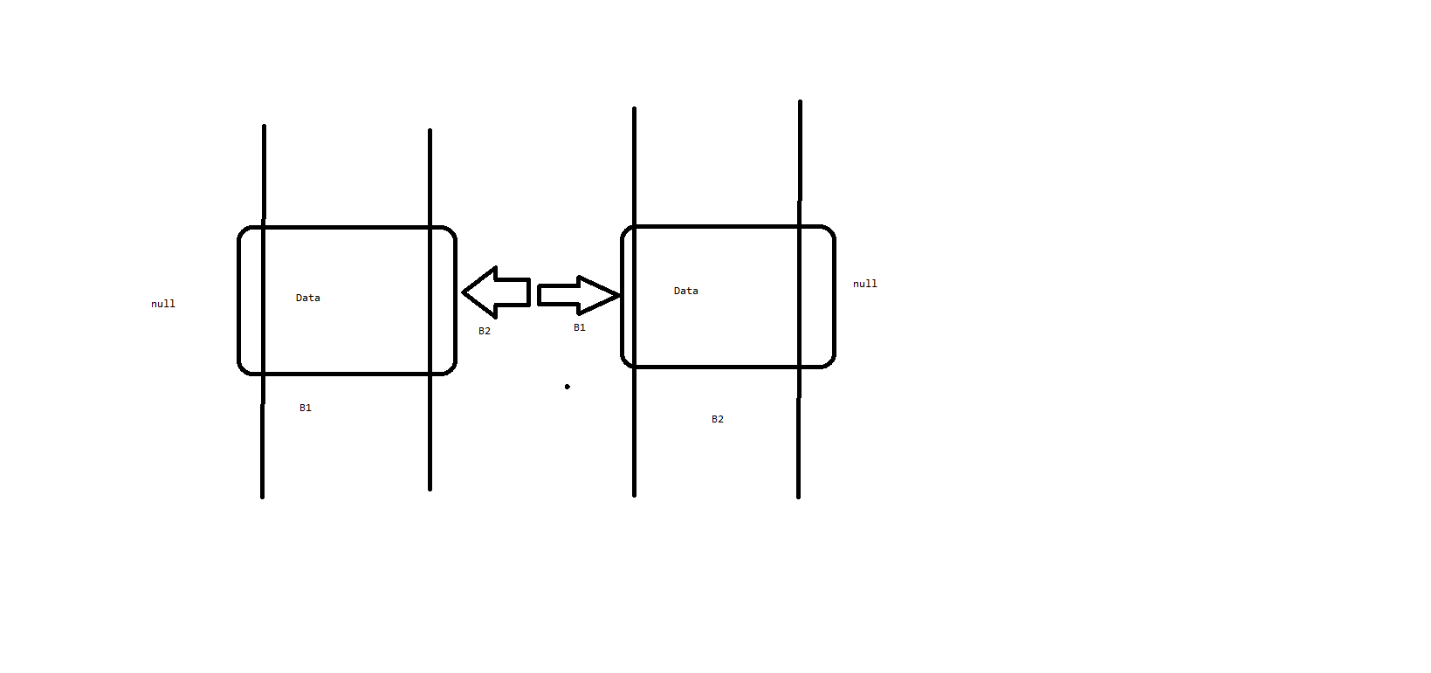
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

al1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
  
\* firstElement(), lastElement(), capacity() functions are not available in array list

Note : Collections.sort() can be used to sort the objects in any collections but of homogeneous data type only.  
\* If we are trying to do this operation for heterogeneous data type then we can expect java.lang.ClassCastException exception.

**Stack – LIFO (Last in first out)**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Stack |  |
| Search | index |  |  |
| 1 | 3 |  | Book4 |
| 2 | 2 |  | Book3 |
| 3 | 1 |  | Book2 |
| 4 | 0 |  | Book1 |

**Linked List – Double linked list  
\*** Linked list memory allocation is not contiguous i.e, linked list memory allocation is exactly like linked list data structure. i.,e one element is linked to another element.  
\* Linked list elements will be scattered in the memory.  
\* Insertion order will be good in linked list because one link will be attached to new element which is inserted.  
\* While writing the code if insertion’s are more then use linked list.  
\* Random access to any element is not good in linked list.  


null Data1 c2 c1 Data2 c3 c2 Data3 null

c1 c4 43 c2 c4 c3

43 56 56

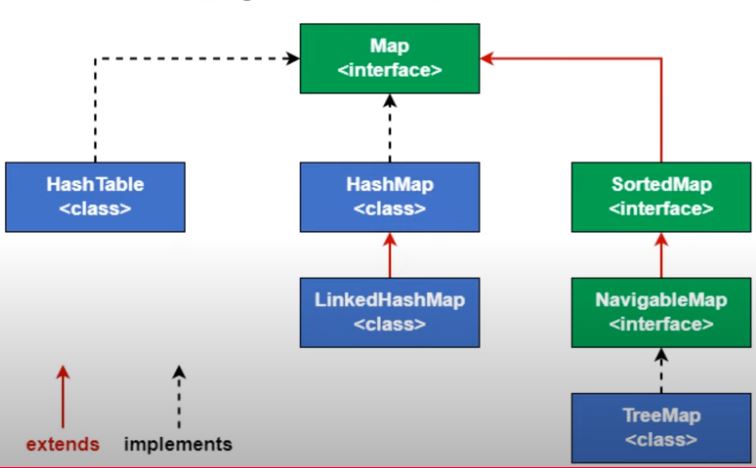
c1 Data4 c3

c4

* If we are trying to remove the object which is not present in the linked list then we can expect - java.util.NoSuchElementException

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \*\*\*\*List  Storage | Array | Array | Array | Double linked list |
|  | **Vector** | **ArrayList** | **Stack** | **Linked List** |
| Default capacity | 10 | 0 | 10 | 0 |
| Initial capacity | 10 | 10 | 10 | 0 |
| Increase capacity | Exact double | Exact half of the size of the array | Exact double | N/A |
| Duplicates | Y | Y | Y | Y |
| null value | Y | Y | Y | Y |
| Insertion order | Y | Y | Y | Y |
| Sorted order | N | N | N | N |
| Random access | Y | Y | Y | Y |
| Special function | firstElement(), lastElement(), capacity() | N/A | peek(),pop(),search(obj),indexOf(obj),empty(),isEmpty() | N/A |
| Synchronized | Y | N | Y | N |
| Good at | Multi threaded program, when ever data increases exponentially | Data storage and retrieval | LIFO | Updating /insertion of data |

**MAP**



**\*** Map memory allocation will be in the form of Key and value pair. **\*** Memory allocation will be random in the map.

Major operations of Map collections are:  
1. Creation of Map  
2. Addition of elements into the Map  
3. Retrieval of elements from the Map (Key and values)  
4. Deletion of elements from the Map  
5. Verification of elements from the Map (Key and values)  
6. Updating of elements in the Map

Hash Table and Has Map  
\* Values will be stored in the form of hash code in the buckets under the heap memory.  
Hash code – “Student1” -> Hashing mechanism - > jfbdfdv233435(Hash code)

100 jfbdfdv233435  
200 fdfdfgf456566

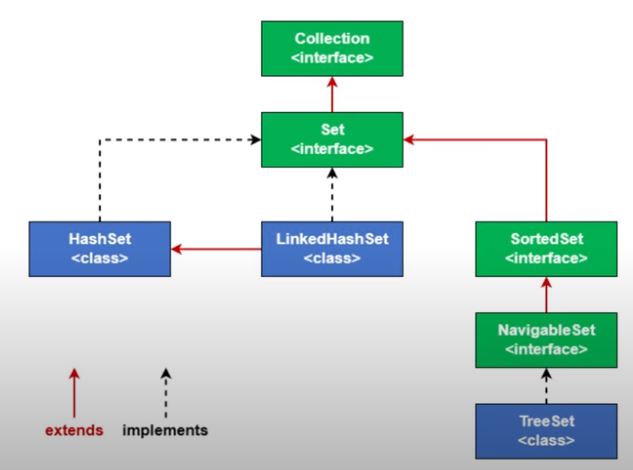
|  |  |
| --- | --- |
| Hashtable | |
| key | value |
| Key1 | value1 |
| Key2 | value2 |
| Key3 | value3 |
| Key4 | value4 |
| Key5 | value5 |
| Key6 | value6 |
| Key7 | value7 |
| Key8 | value8 |
| Key9 | value9 |
| Key10 | value10 |

|  |
| --- |
| Lock |
| key1 |
| value1 |

\* Searching mechanism is not good in map because table will be divided into 2 parts while searching the data. If Key found within in the first half then it will return the value if not it will search for keys in reverse order in the second half of the table.  
  
Ref to day 38 Prog3  
  
Functions related to the map:  
keyset()  
values()  
get(key)  
entrySet()  
getKey()  
getValue()  
remove(key)  
remove(value)  
containsKey(Key)  
containsValue(value)  
put(Key,value)  
replace(Key,Value)  
putIfAbsent(Key,Value)  
clear()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Hashtable | HashMap | LinkedHashMap | Tree Map |
| Default capacity | 0 | 0 | 16 | 0 |
| Initial capacity | 16 | 16 | 16 | 0 |
| Increase capacity | Double | Double | Double | N/A |
| null key | 1 null key can be allowed | 1 null key can be allowed | 1 null key can be allowed | not alowed |
| Duplicates | Duplicate keys are not allowed, but duplicate values will be allowed | Duplicate keys are not allowed, but duplicate values will be allowed | Duplicate keys are not allowed, but duplicate values will be allowed | Duplicate keys are not allowed, but duplicate values will be allowed |
| Insertion order | Random | Random | Sorted according to the insertion order | Sorted |
| Synchronized | N | N | N | N |
| Implementation | Buckets | Buckets | Double linked list | Red-Black tree |
| Interface | Map | Map | Hash Map | Navigable Map, Sorted Map, Map |
| Comments | Absolute | Efficient | Advantage over tree map without extra cost of sorting | Extra cost of sorting the objects |

Ref to day 38 Prog3,4,5,6

**Set:**Major operations of Set collections are:  
1. Creation of Set  
2. Addition of elements into the Set  
3. Retrieval of elements from the Set   
4. Deletion of elements from the Set  
5. Verification of elements from the Set   


\* Hash set internally implements Hash Map  
\* Linked Hash Set internally implements Linked Hash Map  
\* Tree Set internally implements Tree Map  
  
Note: Set collection is used to store only unique elements no duplicates are allowed.

Functions related to the set:  
add(obj)  
size()  
remove(obj)  
removeAll(obj) or clear()  
contains(obj)

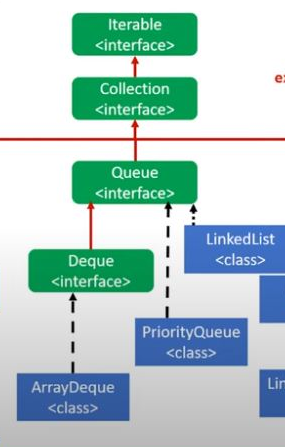
Additional functions for TreeSet  
first() – retrieve the first object  
last() – retrieve the last object  
pollFirst() -– remove the first object  
pollLast() – remove the last object  
subSet(starting range, ending range)

|  |  |  |
| --- | --- | --- |
| Hash Set |  |  |
| key |  | value |
| Key1 |  | value1 |
| Key2 |  | value1 |
| Key3 |  | value1 |
| Key4 |  | value1 |
| Key5 |  | value1 |
| Key6 |  | value1 |
| Key7 |  | value1 |
| Key8 |  | value1 |
| Key9 |  | value1 |
| Key10 |  | value1 |

|  |  |
| --- | --- |
| Set | new Object(); |
| key | value |
| '10 | [java.lang.Object@6d3af739](mailto:java.lang.Object@6d3af739) |
| 20 | java.lang.Object@6d3af739 |
| 30 | java.lang.Object@6d3af739 |
| 40 | java.lang.Object@6d3af739 |

|  |  |  |  |
| --- | --- | --- | --- |
| Set |  |  |  |
|  | HashSet | LinkedHashSet | Tree Set |
| Default capacity | 0 | 16 | 0 |
| Initial capacity | 16 | 16 | 0 |
| Allow duplicates | No | No | No |
| Allow null values | 1 null values is allowed | 1 null values is allowed | not allowed |
| Insertion order | Random/No insertion order | Random/No insertion order | No |
| Sorted order | No | No | YES |
| Random access | No | No | No |
| Additional functions | N/A | N/A | first() ,last() , pollFirst() ,pollLast() , subSet(starting range, ending range) |
| Synchronized | No | No | No |

**Queue**



Major operations of Queue collections are:  
1. Creation of Queue  
2. Addition of elements into the Queue  
3. Retrieval of elements from the Queue  
4. Deletion of elements from the Queue  
5. Verification of elements from the Queue

Priority Queue (Internally implements array)

* No insertion order maintained, objects will be stored randomly in the array.
* Head pointer will be moved according to the priority (lowest number will get the highest priority)
* For getting this expected output use while loop instead of for each loop and use pool() function.  
    
  Ref day 40 : Prog1 and 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PriorityQueue | |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 5 | 2 | 22 | 55 |  |  |  |  |  |  |  | Size =11 |
| Head |  |  |  |  |  |  |  |  |  | Tail |  |  |
| pointer |  |  |  |  |  |  |  |  |  |  |  |  |

\* Always lowest number will get the highest priority.  
Functions related to the Priority Queue:  
Addition:  
add(obj)  
offer(obj)  
  
Retrieval:  
peek()  
  
Deletion:  
poll()  
remove() or clear()  
remove(obj)  
  
verification:  
contains(obj)  
  
Note : Comparator.*reverseOrder*() is used to give the priority in reverse order i.e, head will give priority to the highest number first.

Ref day 40 : Prog 3

**Array Deque (FIFO – First in First out) – double sided array**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array DeQueue** | |  |  |  |  |  |  |  |  |  |  |  |
| **1** | **5** | **2** | **22** | **55** |  |  |  |  |  |  |  | **17** |
| **Head** |  |  |  |  |  |  |  |  |  | **Tail** |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Insertion** |  |  |  |  |  |  |  |  |  | **Insertion** |  |  |
| **Deletion** |  |  |  |  |  |  |  |  |  | **Deletion** |  |  |

Functions related to the Array DeQueue:  
Addition:  
add(obj)  
offer(obj)  
addFirst(obj)  
addLast(obj)  
offerFirst(obj)  
offerLast(obj)  
  
Retrieval:  
peek()  
peekFirst()  
peekLast()

Deletion:  
poll()  
pollFirst()  
pollLast()

remove() or clear()  
remove(obj)  
  
verification:  
contains(obj)  
 **Linked List:**Functions related to the Linked list:

Addition:  
add(obj)

Retrieval:  
peek()

Deletion:  
poll()

verification:  
contains(obj)

|  |  |  |  |
| --- | --- | --- | --- |
| **Queue** |  |  |  |
|  | **Priority queue** | **Array Deque** | **Linked List** |
| **Default capacity** | **11** | **17** | **0** |
| **Initial capacity** | **11** | **17** | **0** |
| **Allow duplicates** | **YES** | **YES** | **YES** |
| **Allow null** | **No** | **No** | **YES** |
| **Insertion order** | **No** | **YES** | **YES** |
| **Sorted order** | **No** | **No** | **No** |
| **Random access** | **No** | **No** | **No** |
| **Synchronized** | **No** | **No** | **No** |
| **Good at** | **Priority based element retrieval** | **FIFO with 2 sided operation** | **Updating the data** |