Absolute Final Backup

INDEX:

CHAPTER 1: REVISION OF CLASS 9 SYLLABUS

1) What is Java:
Java is a popular object oriented programming language that is used to build secure and powerful applications that run across multiple operating systems. The Java language offers flexibility, scalability and maintainability. It was developed by James Gostling and Patrick Naughton for Sun Microsystems in 1991. It was later acquired by Oracle.

2) What is OOP
OOP stands for Object Oriented Programming Paradigm. It views a problem in terms of objects and classes to find the solution.

3) What is an object
An object is an entity with a specific identity, characteristics and behaviour.

4) What is a class
A class is a blueprint representing the set of objects that share common characteristics and behaviour.

6) Name and describe all OOPs principles with examples

- Data Abstraction - It refers to the act of representing essential features without showing background details. Example - To use a switch board, we just need to press certain switches without knowing the internal circuitry of it.

- Encapsulation - The wrapping up of data and functions into a single unit called class is known as encapsulation. Encapsulation hides the details of implementation of an object. It also enables access restriction using access modifiers.

- Inheritance - Inheritance is a mechanism in which one class acquires the property of a pre-existing class . For example, a child inherits the traits of his/her parents. The class inheriting the features is called the sub class/child class/derived class whereas the class whose data members are inherited is called the super class/base class/parent class . The main advantage of Inheritance is Reusability.

- Polymorphism - The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Polymorphism is implemented by Function Overloading. (When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments.)

- Modularity - It is the act of partitioning the program into individual components.

NOTE: Modularity is not an exclusive feature of OOP and if asked to write the principles of OOP, write any of the other principles except Modularity.

7) What is the Character set used by Java?

A character set defines the valid characters that can be used in source programs or interpreted when a program is running.

Java uses *Unicode*, which is a *two -byte character code* and has characters representing almost all human alphabets including all International Languages of the world. A character in Java takes two bytes, as compared to one byte in most other programming languages.

8) What are comments?

Comments are explanatory text that are only for the programmer and are not a part of the code itself , hence not executed by the compiler .There are two kinds of comments in Java -

- Single lined comments - These follow // . Example -
  System.out.print("hello"); // This statement prints 'hello'.
- Multi-line comments - These are enclosed in /* ..... */ and they can span over multiple lines . Example -
  /*This is the start of
  * a new
  * Program
  */

9) What are Tokens?

Tokens are the smallest individual units of a program .There are five kinds of tokens in Java-

1. Keywords - Keywords are words that convey a special meaning to the compiler. Example - public , static , void , import , final , new , true , false etc.

2. Identifiers - Identifiers are names given by the Programmer to the various program units of Java such as the names of variables ,methods ,classes etc.

   Rules for giving Identifiers-

- Identifiers can only have alphabets, digits ,underscore(_) and $ characters and can be of any length.

- They cannot be a Boolean literal, null literal or keyword.
- They cannot begin with a digit.
- They are case sensitive, i.e. My_Variable ≠ my_variable ≠ MY_VARIABLE

3. Literals - Literals are data items that are fixed data values. In simple terms they are the actual values that the programmer assigns to the variables.

There are several kinds of literals-

- *Integer literals* - Integers that can be in decimal form(starts with non-zero number) ,octal form (starts with zero, ex- 014) and hexadecimal form (starts with 0x, ex- 0x5)
- *Floating point literals* - Numbers with decimal point. Can be in fractional form or decimal form.
- *Boolean literals* - There are only two Boolean literals – *true* and *false*.
- *Character literals* – A single character or escape sequence , enclosed in single quotes. Example – 'g' , 'D' , '\t' , '3' , '%' etc.
- *String Literals* – One or more characters enclosed in double quotes. Example – "hello World", "1387586h" , "$hi" etc.
- *Null literal* – It represents an empty/null value and is written as *null.*

4. Operators -

CHAPTER 2: CLASS AS A BASIS OF COMPUTATION

1) What is a object, class and why is class the basis of computation

- An object is an entity with a specific identity, characteristics and behaviour.
- A class is a blueprint of a set of objects that have a common structure and common behaviour.
- All Java programs consist of objects (data and behaviour) that interact with each other by calling methods. All data is stored in objects which are instances of a class. Without classes, there are no objects, and without objects no computation can take place. Thus classes form the basis of all computation in Java.

CHAPTER 3: USER DEFINED FUNCTIONS

1) What is a function

A function is a block of interrelated statements that is only executed when called.

2) Explain types of functions

(a) Void type - They don't have any data type but are prefixed with void. They do not return a value after the function executes.
(b) Return type - They have a return statement. They have a data type which is the same as the type of value returned.

3) Why should we use functions?

(a) Hides complexity - When the program becomes too complex, to cope with the complexity, divide and conquer algorithm is used. Using functions you can divide the program into small modules which are executed only when called.
(b) Reusability - Once a method is defined it can be reused in other classes. Reusability is an important concept in OOPs, which means write once and use many.
(c) Hiding details - We can use a method as a black box and accept the results without concern for the details.

4) What is an access specifier
Is it a keyword used to determine the type of access to a function (It can also be called access modifier). There are four access specifiers -

(a) Public - Public denotes that a variable or a method can be used within its own class, within its own package or it can be used for classes outside the package. Basically, everywhere.
(b) Private - Private denotes that a variable or method can be accessed only within its own class, where it has been declared.
(c) Protected - Protected denotes that a variable or method can be accessed only in its subclasses and within the same package.
(d) Default (NOT A KEYWORD) - When any access specifier is not mentioned. The variable or method can be used only within its own package.

Specifier	Class	Package	Subclass	Outside package
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

5) What is the prototype and the signature

(a) The function prototype is the first line of the function definition that tells the program about the type of value returned by the method and the number and type of arguments.

(b) The signature refers to the number or types of arguments in the prototype.

public int sum (int a, int b) // The whole thing is the prototype. The thing in the bracket is the signature

6) Difference between implicit and explicit conversion

Implicit Conversion	Explicit Conversion
Performed by compiler without programmer's	User defined conversion that forces an

intervention	expression to be of a particular type
Also called widening and coercion	Also called narrowing and type casting

7) Difference between pure and impure methods

Pure methods	Impure Methods
They don't modify the state of arguments received.	They modify the state of arguments received.
Used in call by value	Used in call by reference

8) Difference between Call by Value and Call by Reference (They are types of functions)

Call by Value	Call by Reference
In call by value, a copy of the actual parameters is passed as the formal parameters in the function.	In Call by Reference, the actual address/reference of the actual parameter is sent to the formal parameters in the function.
Any change in the formal parameter is not reflected in the actual parameter	Any change in the formal parameter is reflected in the actual parameter.
All primitive data types (int, double, char etc) are sent using call by value	All Reference data types (objects, strings, arrays) are sent using call by reference

If your English Teacher wanted to test your grammar by writing a grammatically incorrect passage on the board and telling you to **write the same passage again in your notebook, but correct the words while you write, that is call by value.** You made another copy of the data. If she tells you to **erase the incorrect words and write the correct ones on the blackboard itself, that is call by reference**. You did the work on the original data, not copy it separately and doing it.

9) Difference between Formal and Actual Parameter

Formal Parameter	Actual Parameter
They appear in function call	They appear in function definition
A change in the formal parameter doesn't reflect in the actual parameter	A change in the formal parameter in the function body doesn't reflect change in the actual parameter

10) What is dot operator

(.) Dot Operator is used to call methods or instance variables associated with that object.

11) What is function overloading

Function overloading means the same function name, but with different number or types of parameters. The functions have the same name but different signatures. Better definition - A function name having several definitions in the same scope that are differentiable by the number of types of arguments, is said to be an overloaded function.

12) What is a jump statement and explain all types.
In Java jump statements are mainly used to transfer control to another part of our program depending on the conditions. There are three jump statements in Java:

(a) Return - It is used to exit from the current method and control is passed to the calling function. It is also used to return a value to calling code. IMPORTANT: A method may contain several return statements however, only the first one is executed since the method is terminated after the first return statement and control is returned.
(b) Break - It terminates control of current function and passes control to whatever is under it. It can only be used in switch-case, loops and labeled blocks. Outside of these, it is invalid.
(c) Continue - Continue statement is used to skip current iteration and start with next iteration in loops and it can be used to skip the block of statements below it.

CHAPTER 4: CONSTRUCTORS

1) What is a constructor

- It is a special type of function which has the same name as the name of the class.
- It does not have a void or return type.
- It's purpose is to initialize the instance variables.
- It is executed immediately on creation of an object.

2) Name and explain the types of constructors

(a) Default Constructor - It is used to initialize the instance variables to their default values.
(b) Parameterized Constructors - A constructor with parameters is known as a parameterized constructor. It initializes the instance variables to the parameter values.

3) What is a class variable
When a variable is prefixed with the keyword static, it is known as a static data member or class variable. It is called without class object.

Normie methods be like: <objectname>.<methodname> (parameters)
Cool methods that use temporary objects be like: new <classname>().<methodname> (parameters)
Static methods (Da bawse) be like: <methodname> (parameters)

4) Difference between Instance, local and class variables

Instance Variables	Local Variables	Class Variables
It is a variable that is associated with the object.	It is a variable that is defined within a method or block	They are also associated with objects but they are prefixed

		with keyword static. They are also known as static data members.
The scope of these variables is within all functions of the class.	The scope of these variables is within the method or block in which it is declared.	The scope of these variables is throughout the class
Everytime an object is created, a template of the instance variables is assigned to that object. **Each object has their own copy of the instance variables.**	It is not associated with objects.	**Only 1 copy of the variable exists which is shared by every object of the class.**

5) What is the ‘this’ keyword
‘this’ keyword refers to the current calling object for which the method or constructor has ben invoked. It is used when the instance and local variable have the same name. When ‘this’ keyword is not used, the local variable gets preferenced and the instance variable is hidden. We use ‘this’ to specify to Java which variable is the instance and which is local.

Example -

```
class example
{
int a;
int b;
example (int a, int b)
{
        this.a = a; // this.a is the instance variable and a is the local variable
        this.b = b;
}
psvm ()
{
        example (4,5);
}
}
```

CHAPTER 5: LIBRARY CLASSES

1) Explain boxing, autoboxing, unboxing

- Boxing is converting primitive data types to class variables
- Unboxing is converting objects to primitive data types
- AutoBoxing is when the compiler automatically converts primitive types to objects
- AutoUnboxing is when the compiler automatically converts objects to primitive data types

2) Give examples of boxing, autoboxing and unboxing

int a = 69;
Integer autoboxing = a; // Done by compiler automatically
Integer boxing = Integer.valueOf (a); // Manually using wrapper classes
Integer obj = new Integer (a); // Converting primitive to wrapper classes using Constructor of Double Class

Double b = 420;
double unboxing = b; // Automatic

3) What is a wrapper class? List all of them.
Wrapper classes in java provide the mechanism to convert primitive data types into objects and objects into primitive. Types of wrapper Classes:

Primitive Data Types	Wrapper Classes
boolean	Boolean
char	Character
int	Integer
double	Double
byte	Byte
short	Short
long	Long
float	Float

4) Why use wrapper classes in the first place? Why not just use primitive data types
(a) Java is an object oriented language where everything is represented as objects and classes. Using wrapper classes, we can use primitives as objects and use them with all classes.
(b) There are many inbuilt ready-to-use methods which can only be used with objects. Wrapper classes allow us to use them.
(c) Wrapper classes allow primitive data types to be passed by reference, as an argument to a function, if needed.

NOTE: Wrapper classes are final. Their value cannot be changed. This is done to ensure uniform capabilities across all instances.

5) Name and explain all methods of Wrapper Classes
a) static boolean b = Character.isDigit (char ch)
b) static boolean b = Character.isLetter (char ch)
c) static boolean b = Character.isLetterOrDigit (char ch)

d) static boolean b = Character.isLowerCase (char ch)
e) static boolean b = Character.isUpperCase (char ch)
f) static boolean b = Character.isWhiteSpace (char ch)
g) char ch = Character.toUpperCase (char c)
h) char ch = Character.toLowerCase (char c)
i) int i = Integer.parseInt (String str)
j) long L = Long.parseLong (String str)
k) Double d = Double.parseDouble (String str)

6) Define package, libraries and import keyword

- Package - A java package is a collection of classes, interfaces and sub packages.
- Class Libraries - Java includes predefined classes in the form of packages as a part of installation. These packages are called java class libraries. Ex- java.lang, java.util, java.io
- import - Import keyword is used to import built-in and user defined packages into your java source file. Syntax - import <name of package>

CHAPTER 6: ARRAYS

1) What is an array?
An array is a collection of variables of the same type that are referenced by a common name. It is a reference data type. An array is initialized and declared like - int arr[] = new int [5];

2) What are the advantages and disadvantages of using an array?
Advantages -

(a) Arrays represent multiple data items of the same type using a single name.

(b) In arrays, any elements can be accessed randomly by using the index number.

Disadvantages -

(a) Data in arrays is stored in fixed memory locations.

(b) The number of elements to be stored in the array have to be known in advance. The size of arrays cannot be changed once defined.

3) What are the types of arrays
There are two types of arrays - One dimensional arrays and multidimensional arrays consisting of two or more dimensions of arrays.

Note: Indexing of arrays starts with 0 not 1. The first element of arr[] is arr[0], then arr[1] and so on. Also, arr[].length function gives us the number of elements in the array.

4) Explain linear and binary search
Linear search is the searching technique where each element of the array is compared with the search element one by one till the match is found or all elements have been compared.

```
for (int i = 0; i < arr1.length; i++) {
        if (arr1[i] == element) {
        System.out.println ("Element found at " + (i+1) + "th index");
```

```
        flag = 1;
        break;
        }
}
if (flag == 0)
        System.out.println ("Element not found");
```

Binary search is the technique which only works in sorted arrays. The search element is compared with the middle element of the array. If the search element matches the middle element, search finishes. If the search element is less than middle, perform binary search in the first half of the array, otherwise perform binary search in the latter part of the array.

```
int l = 0, u = (arr.length-1);
int m;
while (l <= u) {
        m = (l+u)/2;
        if (element > arr2[m])
           l = m+1;
        else if (element < arr2[m])
           u = m-1;
        else {
           System.out.println ("Element found at " + (m+1));
           flag = 1;
           break;
        }
 }
if (flag == 0)
        System.out.println ("Element not found");
```

6) Difference between Linear and Binary Search

Linear Search	Binary Search
Works in all arrays	Works only in sorted arrays
Linear search finds an element in a list by sequentially checking the elements of the list until finding the matching element.	Binary search is an algorithm that finds the position of a element within a sorted array and only checks in half of the array.
In large arrays, linear search is inefficient.	In large arrays, binary search is more efficient.

7) Explain bubble sort and selection sort *with code*

In ascending Bubble Sort, the adjoining values are compared and exchanged if they are not in order. After one pass, the largest element is put in the end.

```
int temp = 0;
```

```
flag = 0;
for (int i = 0; i < arr2.length; i++) {
	for (int j = 0; j < ((arr2.length)-i-1); j++) {
		if (arr2[j] > arr2 [j+1]) {
			temp = arr2[j];
			arr2[j] = arr2 [j+1];
			arr2[j+1] = temp;
			flag++;
		}
	}
	if (flag == 0)
		break;
	else
		flag = 0;
}
```

In ascending order Selection Sort, the smallest element from the unsorted array is searched for and put in the sorted array. After one pass, the smallest element is put in the starting.

```
int min, temp;
for (int i = 0; i < arr1.length; i++) {
	min = i;
	for (int j = (i+1); j < arr1.length; j++) {
	    if (arr1[j] < arr1[min])
	        min = j;
	}
	temp = arr1[i];
	arr1[i] = arr1[min];
	arr1[min] = temp;
}
```

CHAPTER 7: STRING HANDLING

1) What is a string?

String is a sequence of characters. It is an immutable object in Java. Immutable simply means unmodifiable or unchangeable. Once string object is created its data or state can't be changed but a new string object is created.

2) What is string handling?

String Handling provides a lot of concepts that can be performed on a string such as concatenating string, comparing string, substring etc

3) What happens when we make a string?

Each time you create a string literal, the JVM checks the string constant pool first. If the string already exists in the pool, a reference to the pooled instance returns. If the string does not exist in the pool, a new String object instantiates, then is placed in the pool.

4) Where are string objects stored?
String objects are stored in a special memory area known as string constant pool inside the Heap memory.

5) Why does Java use the concept of string literal?
To make Java more memory efficient (because no new objects are created if it exists already in string constant pool).

6) What is the difference between ==, .equals (), compareTo ()

==	.equals ()	.compareTo ()
The = = operator compares references not values.	equals() method compares the original content of the string	compareTo() method compares strings lexicographically and returns an int which tells if the values. It compares less than, equal, or greater than. Suppose s1 and s2 are two string variables. If: – s1 == s2 :0 – s1 > s2 :positive value – s1 < s2 :negative value
Used in reference matching.	It is used in authentication.	It is used in sorting.

7) Explain all string methods
All methods are stored in java.lang.String class

S No.	Method & Description
1	**char charAt(int index)** Returns the character at the specified index.
2	**int compareTo(Object o)** Compares this String to another Object.
3	**int compareTo(String anotherString)** Compares two strings lexicographically.
4	**int compareToIgnoreCase(String str)**

	Compares two strings lexicographically, ignoring case differences.
5	**String concat(String str)** Concatenates the specified string to the end of this string.
6	**boolean endsWith(String suffix)** Tests if this string ends with the specified suffix.
7	**boolean equals(Object anObject)** Compares this string to the specified object.
8	**boolean equalsIgnoreCase(String anotherString)** Compares this String to another String, ignoring case considerations.
9	**int indexOf(int ch)** Returns the index within this string of the first occurrence of the specified character.
10	**int indexOf(int ch, int fromIndex)** Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
18	**int indexOf(String str)** Returns the index within this string of the first occurrence of the specified substring.
19	**int indexOf(String str, int fromIndex)** Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
21	**int lastIndexOf(int ch)** Returns the index within this string of the last occurrence of the specified character.
22	**int lastIndexOf(int ch, int fromIndex)** Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
23	**int lastIndexOf(String str)** Returns the index within this string of the rightmost occurrence of the specified substring.

24	**int lastIndexOf(String str, int fromIndex)** Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
25	**int length()** Returns the length of this string.
29	**String replace(char oldChar, char newChar)** Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
34	**boolean startsWith(String prefix)** Tests if this string starts with the specified prefix.
35	**boolean startsWith(String prefix, int toffset)** Tests if this string starts with the specified prefix beginning a specified index.
37	**String substring(int beginIndex)** Returns a new string that is a substring of this string.
38	**String substring(int beginIndex, int endIndex)** Returns a new string that is a substring of this string.
39	**char[] toCharArray()** Converts this string to a new character array.
40	**String toLowerCase()** Converts all of the characters in this String to lower case using the rules of the default locale.
42	**String toString()** This object (which is already a string!) is itself returned.
43	**String toUpperCase()** Converts all of the characters in this String to upper case using the rules of the default locale.
45	**String trim()** Returns a copy of the string, with leading and trailing whitespace omitted.

46	**static String valueOf(primitive data type x)** Returns the string representation of the passed data type argument.