Computer Applications Full Syllabus notes -

**INDEX:**

Chapter 1: Revision of class 9 syllabus

Chapter 2: Class as a basis of computation

Chapter 3: User Defined Methods

Chapter 4: Constructors

Chapter 5: Library Classes

Chapter 6: Arrays

Chapter 7: String Handling

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, scalibility 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 byte code?**

Java is a platform independent language. Java achieves this with the help of byte code which is a highly optimized set of instructions executed by the Java Virtual Machine.

What this means is that any source code produced in Java is in the form of byte code and when it is to be used by a machine, it can be converted to native executable code (that can be understood by the machine) by the Java Virtual Machine.

**4) What is an object**

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

**5) 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 Reuseability.
* 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 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 a 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-lined 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?**

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

* Void type - They don’t have any data type but are prefixed with void. They do not return a value after the function executes.
* 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?

* 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.
* 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.
* 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 -

* 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.
* Private - Private denotes that a variable or method can be accessed only within its own class, where it has been declared.
* Protected - Protected denotes that a variable or method can be accessed only in its subclasses and within the same package.
* 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

* 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.
* 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 intervention | User defined conversion that forces an 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 same function name, but with different number or types of parameters. The functions have 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) Explain all jump statements in Java

* Return - It is used to exit from current method and control is passed to 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 method is terminated after the first return statement and control is returned.
* 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.
* 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

* Default Constructor - It is used to initialize the instance variables to their default values.
* 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 been invoked. It is used when the local and instance variable have the same name. In that case, Java hides the instance variable and the local variable gets preference.

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

* 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.
* There are many inbuilt ready-to-use methods which can only be used with objects. Wrapper classes allow us to use them.
* 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.