|  |  |
| --- | --- |
| **What is a file?** | A named location which stores data or information permanently. |
| **What is a class?** | * It is a blueprint from which objects are created. A class contains methods and variables associated with an instance of a class. * Super-class: a class from which other classes inherit. |
| **What is an object?** | It is an instance of a class.The object has state and behaviour. |
| **What is a constructor?** | * It is a method used to create an object of a class. * Two types: default and parameterized constructor. * Default constructor: no arguments or all arguments have default values. |

|  |  |
| --- | --- |
| **C++ v C** | * **C++** is an object-oriented programming paradigm. Organizes software design around data, or objects, rather than functions and logic. * **C is a** structured programming language (programming language divides the problem into smaller structural blocks each of which handles a particular responsibility). |
| **What are the basic OOPs principles?** | OOPLs: object-oriented programming languages.   * Encapsulation * Inheritance * Abstraction * Polymorphysm |
| **Inheritance** | A property in which the property of a parent class (superclass)is passed on to a child class (subclass). |
| **Multiple inheritance** | A process where a subclass can be derived from more than one superclass.   * Advantage: class can inherit functionality of more than one base class. * Disadvantage: can be confusing if two base classes implement methods/variables with same name. |
| **Polymorphysm** | * The ability of an object to take on multiple forms. * Used in OOP: when a language process objects differently depending on their data type or class. * Example: class Shape with method Area; calculates differently for rectangle, circle… |
| **Encapsulation** | * It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. * It is a protective shield that prevents the data from being accessed by the code outside this shield |
| **Abstraction** | We take out unnecessary details and only focus on aspects that are necessary to that context or system under consideration. |
| **What are instance and class variables?** | Instance variable belongs to a particular instance of that class, class variable also known as static variables. |

|  |  |
| --- | --- |
| **Compare method and constructor** | Constructor: used to initialize instance of a class.  Method: used to perform some function or operation. |
| **What is a singleton class?** | It limits the number of objects created for a class to one, but gives flexibility of creating more objects if the situation changes. |
| **What are the steps for creating an object?** | 1. Declared (variable declaration)  2. instantiated (new)  3. Initialization (constructor) |
| **Access modifiers** | 1. No modifier: visible to overall package  2. Private: visible to class only  3. Public: visible to the world  4. Protected: visible to package and subclass |
| **Wrapper class** | Classes to access a primitive data type (boolean, characters, integers) as an object. |
| **Overloading v Overriding** | * Overloading: when two or more methods in the same class have same method name but different parameters * Overriding: two methods having the same method name and parameters, but one of the methods is in the parent class and the other is in the child class |
| **What is a stream?** | * Sequence of data. * Input stream: used to read data from a source. * Output stream: used to write data into destination. |
| **What is an interface?** | * Similar to a class but its collection of abstract methods (no implementation). * A class can implement multiple interfaces. |
| **Class v Interface** | * Interface cannot be instantiated * Interface doesn't have constructors * Interface only has abstract methods * Class implements interface and extends a class * Interface can extend multiple interfaces |
| **Abstract class** | * It may or may not contain abstract methods but, if a class has at least one abstract method, then it must be declared abstract. * Abstract class cannot be instantiated * To use it, we have to inherit it from another class * If we inherit an abstract class, we have to provide implementations to the abstract methods in it |
| **Abstract method** | A method that is declared, but contains no implementation. |
| **Threads** | * Threads are a way for a program to divide (*"split"*) itself into two or more simultaneously (or pseudo-simultaneously) running tasks. * The threads of a process share its executable code and the values of its dynamically allocated variables and non-thread-local global variables at any given time. |
| **Multi-threading** | The ability of a central processing unit (CPU) (or a single core in a multi-core processor) to provide multiple threads of execution concurrently, supported by the operating system. |
| **Deadlock** | A state in which each member of a group is waiting for another member, including itself, to take action |
| **Process** | A process is an executing instance of an application.  A process can contain multiple threads. |
| **Multi-processing** | Multi-processing is the use of is the use of two or more central processing units (CPUs) within a single computer system. |
| **Threads v Processes** | * In most cases, a thread is a component of a process. (Depends on O.S. implementation) * Multiple threads can exist within one process, executing concurrently and sharing resources such as memory, while different processes do not share these resources. * Processes are independent of each other. Threads, since they share the same address are interdependent. Caution must be taken so different threads don’t step on each other. |
| **Multi-treading**  **Pros and Cons** | |  |  | | --- | --- | | Pros | Cons (increased complexity) | | .Better use of CPU resource.  .Enhanced performance by decreased development time.  .Simultaneous and parallelized occurrence of tasks.  .Less maintenance | .Complex debugging and testing.  .Difficulty level in writing a program.  Overhead switching of context.  .Result sometimes unpredictable.  .Likelihood of deadlock. | |
| **Concurrency** | In computer science, concurrency is the ability of different parts or units of a program, algorithm, or problem to be executed out-of-order or in partial order, without affecting the final outcome. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**References**

* + <https://www.multisoftvirtualacademy.com/blog/common-advantages-and-disadvantages-of-multithreading-in-java/>
  + Wikipedia