OOP – Object-Oriented Programming

1. What is Object-Oriented Programming?

Object-Oriented Programming (OOPs) is a type of programming that is based on objects rather than just functions and procedures.

1. Basic OOP concepts:

* Abstraction – simplifying complexity;
* Encapsulation –binding the data and the code that works together in a single unit;
* Inheritance – allows classes to inherit common properties from other classes;
* Polymorphism – refers to the ability to exist in multiple forms. An object takes many forms. A method takes a different quantity of parameters.

1. Why use OOPs?

* OOPs allow clarity in programming thereby allowing simplicity in solving complex problems;
* Code can be reused through inheritance thereby reducing redundancy;
* Data and code are bound together by encapsulation;
* OOPs allow data hiding, therefore, private data is kept confidential;
* Problems can be divided into different parts making them simple to solve;
* The concept of polymorphism gives flexibility to the program by allowing the entities to have multiple forms.

1. Why is OOPs so popular?

OOPs, programming paradigm is considered as a better style of programming. Not only does it help in writing a complex piece of code easily, but it also allows users to handle and maintain them easily as well.

1. What is meant by Structured Programming?

Structured Programming refers to the method of programming which consists of a completely structured control flow. Here structure refers to a block, which contains a set of rules, and has a definitive control flow, such as (if/then/else), (while and for), block structures, and subroutines.

Nearly all programming paradigms include Structured programming, including the OOPs model.

1. What is an object?

* An object is an instance of a class;
* Different objects have different states or attributes, and behaviors.

1. What is a class?

User-defined blueprint from which objects are created. A class is a template that binds the code and data together into a single unit.

1. What is a method?

A callable set of instructions is also called a procedure or function that is to be performed on the given data.

1. What is a Structure?

A structure is a user-defined collection of variables that are made of different data types.

1. Can you call the base class method without creating an instance?

Yes, you can call the base class without instantiating it if:

* It is a static method;
* The base class is inherited by some other subclass.

1. Are there any limitations on Inheritance?

Yes. Inheritance needs more time to process, as it needs to navigate through multiple classes for its implementation. Also, the classes involved in Inheritance - the base class and the child class, are very tightly coupled together. So, if one needs to make some changes, they might need to do nested changes in both classes. Inheritance might be complex for implementation, as well. So, if not correctly implemented, this might lead to unexpected errors or incorrect outputs.

1. What are the different types of inheritance?

* Single inheritance
* Multiple inheritances
* Multilevel inheritance
* Hierarchical inheritance
* Hybrid inheritance

1. What is a superclass?

A superclass or base class is a class that acts as a parent to some other class or classes.

1. What is a subclass?

A class that inherits from another class is called the subclass.

1. What is method overloading?

Two or more methods have the same name but different parameters or signatures. Resolved during compile time.

1. What is method overriding?

Child class redefining methods present in the base class with the same parameters/ signature. Resolved during runtime.

1. Access modifiers:

* Public – gives the possibility to access from everywhere in the program;
* Private – gives the possibility to access only within the class;
* Protected – gives the possibility to access subclasses or within the same package;
* Default – gives the possibility of access only within the package.

1. What is an abstract class?

An abstract class is a class that consists of abstract methods. These methods are declared but not defined. If these methods are to be used in some subclass, they need to be exclusively defined in the subclass. Instances of an abstract class cannot be created because it does not have a complete implementation. However, instances of subclass inheriting the abstract class can be created.

1. What is an interface?

An interface refers to a special type of class, which contains methods, but not their definition. Only the declaration of methods is allowed inside an interface. To use an interface, you cannot create objects. Instead, you need to implement that interface and define the methods for their implementation.

1. What are virtual functions?

Virtual functions are functions that are present in the parent class and are overridden by the subclass. These functions are used to achieve runtime polymorphism.

1. What are pure virtual functions?

Pure virtual functions or abstract functions are functions that are only declared in the base class. This means that they do not contain any definition in the base class and need to be redefined in the subclass.

1. What is a constructor?

A constructor is a special type of method that has the same name as the class and is used to initialize objects of that class.

1. Types of constructors:

* Default constructor
* Parameterized constructor
* Copy constructor – creates objects by copying variables from another object of the same class
* Static constructor
* Private constructor

1. What is a destructor?

A destructor is a method that is automatically invoked when an object is destroyed. The destructor also recovers the heap space that was allocated to the destroyed object, closes the files and database connections of the object, etc.

1. What is Garbage Collection (GC)?

GC is an implementation of automatic memory management. The Garbage collector frees up space occupied by objects that are no longer in existence.

1. What is the use of ‘finalize’?

Finalize as an object method used to free up unmanaged resources and cleanup before Garbage Collection (GC). It performs memory management tasks.

1. What are the different types of garbage collectors in Java?

Garbage collection in Java is a program that helps in implicit memory management. In Java, using the new keyword you can create objects dynamically, which once created will consume some memory. Once the job is done and there are no more references left to the object, Java using garbage collection destroys the object and relieves the memory occupied by it. Java provides four types of garbage collectors:

* Serial Garbage Collector
* Parallel Garbage Collector
* CMS Garbage Collector
* G1 Garbage Collector

1. What is an error?

Error is an irrecoverable condition occurring in runtime.

1. What is an exception?

An exception is a kind of notification that interrupts the normal execution of a program. Exceptions provide a pattern to the error and transfer the error to the exception handler to resolve it. The state of the program is saved as soon as an exception is raised. Conditions that an application might try to catch.

1. What is exception handling?

Exception handling in Object-Oriented Programming is a very important concept that is used to manage errors. An exception handler allows errors to be thrown and caught and implements a centralized mechanism to resolve them.

1. What is a try/ catch block?

A try/ catch block is used to handle exceptions. The try block defines a set of statements that may lead to an error. The catch block catches the exception.

1. What is a finally block?

A finally block consists of code that is used to execute important code such as closing a connection, etc. This block executes when the try block exits. It also makes sure that finally, the block executes even in case some unexpected exception is encountered.

1. What are the limitations of OOPs?

* Usually not suitable for small problems
* Requires intensive testing
* Takes more time to solve the problem
* Requires proper planning
* The programmer should think of solving a problem in terms of objects

1. How much memory does a class occupy?

Classes do not consume any memory. They are just a blueprint based on which objects are created. Now when objects are created, they initialize the class members and methods and therefore consume memory.

1. Is it always necessary to create objects from class?

No. An object is necessary to be created if the base class has non-static methods. But if the class has static methods, then objects don’t need to be created. You can call the class method directly in this case, using the class name.

1. What is Overloading?

Overloading is a compile-time polymorphism feature in which an entity has multiple implementations with the same name.

1. What is Overriding?

Overriding is a runtime polymorphism feature in which an entity has the same name, but its implementation changes during execution.

1. Recursion – a technique to iterate over an operation by having a function call itself repeatedly until it arrives at a result.
2. Coupling is a degree of interdependence between software modules. Low/loose coupling often correlates with high cohesion. It is often thought to be a sign of a well-structured computer system and a good design, and combined with high cohesion, supports the general goals of high readability and maintainability. High/strong coupling is said to occur when one module uses the code of another module for instance branch.
3. Cohesion refers to the degree to which the elements inside a module belong together. Modules with high cohesion are associated with several desirable traits of software including robustness, reliability, reusability, and understandability. Low cohesion is associated with difficulty to maintain, test, reuse, and understand.

OOP – Обектно-ориентирано програмиране

1. Какво представлява обектно-ориентираното програмиране?

Обектно-ориентираното програмиране (OOPs) е вид програмиране, което се основава по-скоро на обекти, а не само на функции и процедури.

2. Основни концепции за OOP:

        Абстракция – опростяване на сложността;

        Капсулиация –обвързване на данните и кода, който работи заедно в една единица;

        Наследяване – позволява на класовете да наследяват общи свойства от други класове;

        Полиморфизъм – се отнася до способността да съществува в множество форми. Обектът приема много форми. Метод отнема различно количество параметри.

3. Защо да използвате OOPs?

        УОП позволяват яснота при програмирането, като по този начин позволяват простота при решаването на сложни проблеми;

        Код може да се използвате повторно чрез наследство по този начин намаляване на съкращенията;

        Данните и кодът са обвързани заедно с капсулира;

        OOPs позволяват скриване на данни, поради което частните данни се пазят поверителни;

        Проблемите могат да бъдат разделени на различни части, които ги правят прости за решаване;

        Концепцията за полиморфизъм дава гъвкавост на програмата, като позволява на субектите да имат множество форми.

4. Защо OOPs е толкова популярен?

OOPs, програмиране парадигма се счита за по-добър стил на програмиране. Не само че помага в писането на сложно парче код лесно, но също така позволява на потребителите да се справят и да ги поддържат лесно, както и.

5. Какво се има предвид със Структурирано програмиране?

Структурираното програмиране се отнася до метода на програмиране, който се състои от напълно структуриран контролен поток. Тук структурата се отнася до блок, който съдържа набор от правила и има окончателен контролен поток, като например (if/then/else), (докато и за), блокови структури и подпрограми.

Почти всички програмни парадигми включват Структурирано програмиране, включително модела на ООП.

6. Какво е обект?

        Обектът е екземпляр на клас;

        Различните обекти имат различни състояния или атрибути, и поведения.

7. Какво е клас?

Дефиниран от потребителя чертеж, от който се създават обекти. Класът е шаблон, който свързва кода и данните заедно в една единица.

8. Какво е метод?

Позивен набор от инструкции също се нарича процедура или функция, която трябва да се извърши на дадените данни.

9. Какво е Структура?

Структура е потребител дефинирана колекция от променливи, които са изработени от различни типове данни.

10. Можете ли да се обадите на метода на базовия клас, без да създавате екземпляр?

Да, можете да се обадите на базовия клас, без да го моментирате, ако:

        Това е статичен метод;

        Базовият клас е наследен от някои други подкласове.

11. Има ли някакви ограничения за Наследството?

Да. Наследяването се нуждае от повече време за обработка, тъй като трябва да се ориентира през множество класове за неговото изпълнение. Също така, класовете, участващи в Наследяване - базовият клас и детският клас, са много плътно съчетани заедно. Така че, ако човек трябва да направи някои промени, може да се наложи да направи вложени промени в двата класа. Наследството може да е сложно и за изпълнение. Така че, ако не е правилно внедрена, това може да доведе до неочаквани грешки или неправилни изходи.

12. Какви са различните видове наследство?

        Единично наследство

        Множество наследствия

        Многостепенно наследство

        Йерархично наследство

        Хибридно наследство

13. Какво е суперклас?

Суперклас или базов клас е клас, който действа като родител на някой друг клас или класове.

14. Какво е подклас?

Клас, който наследява от друг клас, се нарича подкласа.

15. Какво представлява претоварването на метода?

Два или повече метода имат едно и също име, но различни параметри или подписи. Разрешен по време на времето за компил.

16. Какъв метод е императрик?

Дете клас предефиниране методи присъства в базовия клас със същите параметри / подпис. Разрешен по време на изпълнение.

17. Модификатори за достъп:

        Публично – дава възможност за достъп отвсякъде в програмата;

        Private – дава възможност за достъп само в рамките на класа;

        Защитени – дава възможност за достъп до подкласове или в рамките на същия пакет;

        По подразбиране – дава възможност за достъп само в рамките на пакета.

18. Какво е абстрактен клас?

Абстрактен клас е клас, който се състои от абстрактни методи. Тези методи са декларирани, но не са определени. Ако тези методи трябва да се използват в някои подкласове, те трябва да бъдат изключително определени в подкласа. Екземпляри на абстрактен клас не могат да бъдат създадени, защото той няма пълно изпълнение. Въпреки това могат да се създадат екземпляри на подклас, наследяващи абстрактния клас.

19. Какво е интерфейс?

Интерфейс се отнася до специален тип клас, който съдържа методи, но не и тяхното определение. Само декларацията за методи е разрешена вътре в интерфейс. За да използвате интерфейс, не можете да създавате обекти. Вместо това трябва да внедрите този интерфейс и да определите методите за тяхното изпълнение.

20. Какво представляват виртуалните функции?

Виртуалните функции са функции, които присъстват в родителския клас и са надхвърлили от подкласа. Тези функции се използват за постигане на runtime полиморфизъм.

21. Какво представляват чистите виртуални функции?

Чисти виртуални функции или абстрактни функции са функции, които са декларирани само в базовия клас. Това означава, че те не съдържат никакво определение в базовия клас и трябва да бъдат предефинирани в подкласа.

22. Какво е конструктор?

Конструкторът е специален тип метод, който има същото име като класа и се използва за инициализиране на обекти от този клас.

23. Видове конструктори:

        Конструктор по подразбиране

        Параметризиран конструктор

        Копиране на конструктор – създава обекти чрез копиране на променливи от друг обект от същия клас

        Статичен конструктор

        Частен конструктор

24. Какво е разрушител?

Разрушител е метод, който автоматично се извиква, когато обект е унищожен. Разрушителят също така възстановява хеп пространството, което е било разпределено на разрушен обект, затваря файловете и връзките на базата данни на обекта и т.н.

25. Какво представлява Събирането на боклука (GC)?

GC е внедряване на автоматично управление на паметта. Колекторът Боклук освобождава място, заемано от обекти, които вече не съществуват.

26. Каква е употребата на "финализиране"?

Финализиране като обектен метод, използван за освобождаване на незавършени ресурси и почистване преди Събиране на боклука (GC). Тя изпълнява задачи за управление на паметта.

27. Какви са различните видове събирачи на боклук в Java?

Събиране на боклук в Java е програма, която помага в имплицитното управление на паметта. В Java, с помощта на новата ключова дума можете да създавате обекти динамично, които веднъж създадени ще консумират някои памет. След като работата е свършена и няма повече препратки, оставени на обекта, Java с помощта на събиране на боклука унищожава обекта и облекчава паметта, заета от него. Java осигурява четири вида колекционери за боклук:

        Сериен колектор за боклук

        Паралелен колекционер за боклук

        CMS Колекционер за боклук

        G1 Колекционер за боклук

28. Какво е грешка?

Грешката е невъзстановено условие, възникващо по време на изпълнение.

29. Какво е изключение?

Изключение е вид известие, което прекъсва нормалното изпълнение на дадена програма. Изключения предоставят модел на грешката и прехвърляне на грешката на манипулатора на изключения, за да го разрешите. Състоянието на програмата се запазва веднага след повишаване на изключение. Условия, които дадено приложение може да се опита да хване.

30. Какво е обработка на изключенията?

Работа с изключения в Обектно-ориентирано програмиране е много важна концепция, която се използва за управление на грешки. Манипулатор на изключения позволява грешки да бъдат хвърлени и уловени и прилага централизиран механизъм за разрешаването им.

31. Какво е опит / улов блок?

За боравене с изключения се използва блок за пробване/ улов. Блокът за опит определя набор от изявления, които може да доведат до грешка. Блокът за улов хваща изключението.

32. Какво е най-накрая блок?

Най-накрая блок се състои от код, който се използва за изпълнение на важен код като затваряне на връзка и т.н. Този блок се изпълнява, когато опитът блокира изходи. Също така се уверява, че накрая блокът изпълнява дори в случай, че се срещне някакво неочаквано изключение.

33. Какви са ограниченията на УОП?

        Обикновено не е подходящ за малки проблеми

        Изисква интензивно тестване

        Отнема повече време за решаването на проблема

        Изисква правилно планиране

        Програмистът трябва да мисли за решаване на проблем по отношение на обекти

34. Колко памет заема даден клас?

Класовете не консумират никаква памет. Те са просто чертеж, базиран на това кои обекти са създадени. Сега, когато се създават обекти, те инициализират членовете и методите на класа и следователно консумират памет.

35. Винаги ли е необходимо да се създават обекти от клас?

Не. Обект е необходимо да се създаде, ако базовият клас има нестатични методи. Но ако класът има статични методи, тогава обектите няма нужда да се създават. Можете да се обадите директно на метода на класа в този случай, като използвате името на класа.

36. Какво представлява претоварването?

Претоварването е съставно-време полиморфизъм функция, в която предприятието има множество внедрявания с едно и също име.

37. Какво представлява Overriding?

Overriding е runtime полиморфизъм функция, в която един обект има същото име, но изпълнението му се променя по време на изпълнение.

38. Рекурсия – техника за итерес над операция, като има функция се обажда многократно, докато пристигне в резултат.

39. Прикачването е степен на взаимозависимост между софтуерните модули. Ниското/хлабаво прикачване често корелира с високо сближаване. Често се смята за знак за добре структурирана компютърна система и добър дизайн, и в съчетание с високо сближаване, подкрепя общите цели за висока четливост и поддръжка. Високо/силно съединител се казва, че възниква, когато един модул използва кода на друг модул за инстанция клон.

40. Сближаването се отнася до степента, в която елементите вътре в даден модул принадлежат заедно. Модулите с високо сближаване са свързани с няколко желани черти на софтуера, включително здравина, надеждност, многократна експлоатация и разбираемост. Ниското сближаване е свързано с трудност да се поддържа, тества, използва повторно и разбира.