Въпроси за интервю – ООП

1. **Какво е ООП? - Модел за компютърно програмиране, което организира софтуерен дизайн около обекти**
    - **Обект – множество данни, инстанция на клас**
    - **Клас – blueprint (чертеж) за обекти**
    - **Придимство на ООП е – лесното troubleshooting, преизползването на код, гъвкавост чрез полиформизъм**
2. **Кои са модификаторите за достъп?**

Modifier	Class	Package	Subclass	World
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
No modifier	Yes	Yes	No	No
Private	Yes	No	No	No

3. **Кои са принципите на ООП?**
    - **Абстракция – означава да се ограничим само до съществените характеристики на даден обект, които изпълняват дадена задача и да се абстрахираме (пропуснем) несъществените за задачата характеристики || означава да работим с нещо, което знам как да използваме, без да знаем как работи вътрешно, всяка конкретна имплементация на поведение е скрита в своя обект, за външния свят е видимо само поведението**
    - **Енкапсулация – контролира достъпа до споделените данни, с помощта на модификатори за достъп**
    - **Наследяване – позволява преизползване и разширяване на състояние и поведение на вече съществуващи класове**
    - **Полиморфизъм – способносста на даден обкет да се държи като инстанция на друг клас или като имплементация на друг интерфейс, постига се чрез:**
        - **Overriding – класът-наследник предефинира поведението на класа-родител**
        - **Overloading – класът деклалира методи с едно и също име и различен брой и/или тип параметри**

4. **Каква е разликата между overloading и overriding?**

	Overloading	**Overriding**
Кога	Compile-time	Runtime
Къде	В същия клас	В класове-деца
Runtime performance	По-добро	
Return type	Може да са различни	Запазва се
Static, private & final methods	Да	Не
Binding	Статично	Динамично
Списък от аргументи	Различен	Запазва се

5. **Какво действие има ключовата дума this? – Референция към конкретния обект, достъпване на член-променливи, извикване на констуктор, извикване на произволен метод**
6. **Какво действие има ключовата дума super? – Референция към прекия родител на обекта, достъпване на член-данни на родителя, извикване на констуктор на родителя, извикване на произволен метод, достигаме само public/protected данни**
7. **Какво действие има ключовата дума final?**
    - **В декларацията на променлива, прави я константа**
    - **В декларацията на метод, методът не може да бъде override**
    - **В декларацията на клас, класът не може да се наследява**
8. **Какво действие има ключвоата дума static? – Статичните променливи и методи са част от класа, а не от конкретната инстанция и могат да бъдат достъпени без да е създаден обект**
    - **Статични член-данни – имат единствено копие, което се споделя от всички инстанции на класа**
    - **Статични методи – имат достъп само статични член-данни и други статични методи на класа**
9. **Какво е пакет? – Именувани групи от семантично свързани класове, служат за йерархично организиране на кода, съответстват на файлова система, достъп до клас от друг пакет чрез import, “com.google.mail”**
10. **Каква е йерархията на класовете в Java? – Всички класове в Java са наследните на класа java.lang.Object и може да override методите – boolean equals(Object o), int hashCode(), String toString(), Object clone()**

11. Какво е абстрактен клас? – Дефинират се с модификатора ‘abstract’

- **Могат да имат методи без имплементация, които се дефинират с помощта на ‘abstract’**
- **Не се напълно дефинирани (остават на наседниците да ги допълнят)**
- **Един клас не може да бъде едновременно final & abstract, защото final ограничава класа да бъде наследен**
- **Не могат да се създават обекти от тях**

12. Какво е интерфейс? – Съвкупност от декларация на методи без имплементация

- **Описват формално поведение, без да го имплементират**
- **Може да съдържа static final член-данни -> константи**
- **Може да съдържа default и static методи с имплементация (J8)**
- **Може да съдържа private методи с имплементация (Java 9)**
- **Методите на интерфейсите са public & abstract по подразбиране**
- **Методите не могат да бъдат final**
- **Интерфейсите не могат да се инстанцират**
- **Можем да присвояваме инстанция на клас на променлива от тип интерфейс, който класът имплементира**
- **Можем да проверяваме с instanceof дали клас имплементира даден интерфейс**
- **Ако даден клас декларира, че имплементира интерфейс, той трябва или:**
  - **Да даде дефиниции на всичките му методи**
  - **Да бъде деклариран като абстрактен**
- **Default-ен метод е метод, който има имплементация, но може да бъде override-нат (Благодарение на него, не е нужно да предефинираме нов добавен метод във всеки клас, който имплементира интерфейса)**
- **Private методи използваме, когато имаме два или повече default-ни или статични методи, чиято имплементрация частично се дублира**

13. Какво е Enum class? – Специален референтен тип (клас), предоставящ фиксирано множество от инстанции-константи, не може наследява

14. Каква е разликата между интерфейс и абстрактен клас?

Abstract class	**Interface**
Може да има абстрактни и неабстрактни методи	Може да има абстрактни, default, private и статични методи
Не подържа множествено наследяване	Подържа множествено наследяване
Може да имплементира интерфейс	Не може да наследи клас
Може да има private & protected член-данни	Може да има единствено public член-данни

15. Каква е разликата между структура и клас?

Class	**Struct**
Pass-by-reference	Pass-by-value
Намериа се в Heap паметта	Намира се в Stack паметта
Позводят cleanup (garbage collector)	Не позвовя cleanup (no efficient memory management)
Всички член-данни са private по подразбиране	Всички член-данни са public по подразбиране
Подържа наследяване	Не подържа наследяване

16. Кой са SOLID принципите?

- **Single Responsibility – клас трябва да има само едно задължение**
- **Open/Closed – ентитата на класа трябва да бъдат отворени за допълнение, но затворени за модификация**
- **Liskov Substitution – обекти в програмата могат да бъдат заменени с техни съб-типове, без да нарушават правилността на прогрмата**
- **Interface Segregation – клиентът не трябва да бъде длъжен да имплементира интерфейси, които не използва**
- **Dependency Inversion – Висок „левел“ модули не трябва да зависят от нисък „левел“ модули, а трябва да зависят от абстракция**

17. Как да направим един клас ‘immutable’?

- **Правим класът final, за да не може да бъде наследен**
- **Правим всички атрибюти final**

- **Не прилагаме сетъри**
- **Всички mutable полета, трябва да бъдат final**
- **Клининг на обекти се извръща в гетъра**

18. Кои са 'Design Patterns'?

- **Creational Pattern – осигуряват начин да се създават обекти на класове, скривайки логиката по създаването им (вместо да се инстанцират директно чрез оператора new)**
  - **Factory – използваме, когато имаме родителски клас с няколко наследници и искаме да създаваме един от наследните на родителския клас според параметър**
  - **Builder – решава някой проблеми на Factory за класове с много атрибюти, от които има optional**
  - **Singleton – клас, който има единствена инстанция и може да се достъпва от всяка част на прогрмата**
- **Structural Pattern – осигуряват различен начин за създаване на по-сложни класове чрез наследяване и композиция на по-прости**
  - **Flyweight – позволява да се съберат повече обекти в наличната памет чрез споделяне на общите части**
- **Behavioral Pattern – свързани са с комуникацията между обекти**
  - **Iterator – позволява последователното обхождане на поредица от обекти**
  - **Command – използва се за имплементиране на loose coupling в модел тип заявка – отговор**
  - **Observer – удобен е, когато се интересуваме от състоянието на даден обект и искаме да бъдем нотифицирани, когато има промяна в състоянието му**

19. Какво е конструктор? – Блок от код, който се използва, за да се инстанцилира обект

- **Default Constructor**
- **Parameterized Constructor**