Priraďte

1. Priraďte, každému návrhovému vzoru kategóriu do ktorej patrí.

Abstract Factory	> Creational
Adapter	> Structural
Bridge	> Structural
Builder	> Creational
Chain of Responsibility	> Behavioral
Command	> Behavioral
Composite	> Structural
Decorator	> Structural
Facade	> Structural
Factory Method	> Creational
Flyweight	> Structural
Interpreter	> Behavioral
Iterator	> Behavioral
Mediator	> Behavioral
Memento	> Behavioral
Observer	> Behavioral
Prototype	> Creational
Proxy	> Structural
Singleton	> Creational
State	> Behavioral
Strategy	> Behavioral
Template Method	> Behavioral
Visitor	> Behavioral

Čo je zámerom

2. Čo je zámerom návrhového vzoru Abstract Factory?

Poskytnúť rozhranie na vytváranie objektov viacerých tried, pričom voľbu konkrétnych tried necháte na implementáciu

3. Čo je zámerom návrhového vzoru Adapter?

Zabaliť existujúci objekt do nového rozhrania

Prispôsobiť rozhranie existujúceho objektu potrebám klienta

4. Čo je zámerom návrhového vzoru Bridge?

Umožniť voľbu implementácie nezávisle od voľby abstrakcie/rozhrania

Oddeliť abstrakciu od implementácie

5. Čo je zámerom návrhového vzoru Builder?

Oddeliť vytváranie komplexných objektov od ich reprezentácie (detailnej špecifikácie)

6. Čo je zámerom návrhového vzoru Chain of Responsibility?

Umožniť klientovi odoslať príkaz aj bez toho, aby vedel kto ho vykoná

7. Čo je zámerom návrhového vzoru Composite?

Poskytnúť jednotné rozhranie pre prácu so samostatnými objektami aj kontainerom.

Reprezentovať komplexný objekt ako stromovú štruktúru.

8. Čo je zámerom návrhového vzoru Command?

Umožniť narábať s operáciou ako objektom

Zabaliť príkaz do objektu

9. Čo je zámerom návrhového vzoru Observer?

Poskytnúť možnosť reakcie na udalosť/informáciu viacerým objektom

10. Čo je zámerom návrhového vzoru Prototype?

Vytvoriť nový objekt kopírovaním.

11. Čo je zámerom návrhového vzoru Proxy?

Vytvoriť prostredníka, ktorý bude umožňovať prístup k objektu

Poskytnúť zástupcu, ktorý rezervuje miesto pre skutočný objekt

Poskytnúť prostredníka, ktorý bude kontrolovať prístup k objektu

12. Čo je zámerom návrhového vzoru Singleton?

Zabezpečiť, že bude vytvorená jediná inštancia triedy

13. Čo je zámerom návrhového vzoru Visitor?

Poskytnúť operáciu pracujúcu s objektami rôznych typov (tvoriacich zložitejšiu štruktúru)

14. Čo je zámerom návrhového vzoru Decorator?

Pridať funkcionalitu objektu bez nutnosti vytvoriť podtriedu

15. Čo je zámerom návrhového vzoru Template Method?

Vytvoriť kostru algoritmu a detaily prenechať na podtriedy

16. Čo je zámerom návrhového vzoru Factory Method?

Poskytnúť rozhranie na tvorbu objektu, pričom rozhodnutie aký objekt sa vytvorí necháte na implementáciu

Aký návrhový vzor

17. Potrebujete počítať prístupy k objektu. Aký návrhový vzor by ste použili?

Proxy

18. Potrebujete mať možnosť výberu implementácie aj rozhrania komponenty nezávisle na sebe. Aký návrhový vzor by ste použili?

Bridge

19. Potrebujete implementovať komunikačnú architektúru PUBLISH-SUBSCRIBE. Aký návrhový vzor by ste použili?

Observer

20. Potrebujete rozšíriť funkcionalitu triedy bez použitia podtried. Aký návrhový vzor by ste použili?

Decorator

21. Potrebujete zabezpečiť aby globálny zdielaný prístupový bod k databáze/mailovému serveru/window- manageru... Aký návrhový vzor by ste použili?

Singleton

22. Potrebujete zabezpečiť aby existovala len jediná inštancia vašej triedy. Aký návrhový vzor by ste použili?

Singleton

23. Potrebujete vytvárať objekty pričom ich vytváranie je veľmi náročné na čas a/alebo zdroje. Aký návrhový vzor by ste použili?

Prototype

24. Potrebujete kontrolovať, kto má prístup k objektu. Aký návrhový vzor by ste použili?

Proxy

25. Komponenta, ktorú chcete použiť, nemá rozhranie vyhovujúce vašim dátovým objektom. Aký návrhový vzor by ste použili?

Adapter

26. Pri implementácii aplikácie/frameworku viete, kedy sa vytvára inštancia istého objektu, nepoznáte však ešte konkrétnu triedu. Aký návrhový vzor by ste použili?

Factory Method

27. Pri implementácii aplikácie / frameworku viete, kedy sa vytvára inštancia istého objektu, nepoznáte však ešte konkrétnu triedu. Aký návrhový vzor by ste použili?

Factory Method

28. Udalosťami riadená aplikácia potrebuje poskytnúť UNDO podporu pre akcie. Aký návrhový vzor by ste pri tom využili?

Command

29. Požiadavka nemôže byť spracovaná hneď ako bola vygenerovaná, ale treba čakať na vhodný okamih. Aký návrhový vzor by ste použili?

Command

30. Implementujete GUI framework, ktorý má podporovať viaceré look-and-feel a témy. Aký návrhový vzor by ste tu využili?

Abstract Factory

Ktorý návrhový

31. Pre ktorý návrhový vzor je charakteristická metóda accept()?

Visitor

32. Ktorý návrhový vzor môže pri svojej implementácii využiť wrapper?

Adapter

Decorator

33. Pre ktorý návrhový vzor je charakteristická metóda getInstance()?

Singleton

34. Pre ktorý návrhový vzor je charakteristická metóda execute()?

Command

35. Pre ktorý návrhový vzor je charakteristická metóda clone()?

Prototype

36. Pre ktorý návrhový vzor sú charakteristické metódy attache() a detache()?

Observer

37. Pre ktorý návrhový vzor sú charakteristické metódy update() a notify()?

Observer

38. Pre ktorý návrhový vzor sú typické komponenty Abstraction a Implementor?

Bridge

39. Ktorý návrhový vzor je alternatívou pre objektovo-orientovaný callback?

Command

40. Ktorý návrhový vzor obsahuje rekurzívnu štruktúru?

Composite

Chain of Responsibility

41. Ktorý návrhový vzor je alternatívou pre statický objekt?

Singleton

1-2 vetami

42. Stručne (max. 1-2 vetami) porovnajte návrhové vzory Adapter a Decorator.

Adaptér je určený na zmenu rozhrania existujúceho objektu. Dekorátor vylepšuje iný objekt bez zmeny jeho rozhrania.

43. Stručne (max. 1-2 vetami) porovnajte návrhové vzory Adapter a Bridge.

Bridge: oddeluje abstrakciu od implementacie, aby sa mohli lisit a riesit nezavisle na sebe, toto necha kod klienta nezmeneny
Adapter: konvertuje interface do iného kompatibilného interfacu

44. Stručne (max. 1-2 vetami) vysvetlite, ako súvisia vzory Command a Chain of Responsibility.

Handler-i v Chain of Responsibility môžu byť implementované ako Command-y. Oba vykonávajú príkazy, pričom nevedia nič o prijímateľovi a nepoznajú detaily operácie

45. Stručne (max. 1-2 vetami) porovnajte návrhové vzory Proxy a Adapter.

Proxy pouziva pre vsetky objekty rovnaky interface (wrapper). Adapter konvertuje existujuci interface na odlisny, aby docielil kompatibilitu s inym rozhranim

46. Stručne (max. 1-2 vetami) vysvetlite, čím sa líšia a čo majú spoločné návrhové vzory Builder a Abstract Factory.

Builder sa zameriava na konštrukciu zložitejších objektov krok po kroku. Abstract Factory sa špecializuje na vytváranie skupín príbuzných objektov. AF vráti inštanciu okamžite, zatiaľ čo Builder umožní pred načítaním inštancie vykonať niekoľko ďalších konštrukčných krokov.

47. Stručne (max. 1-2 vetami) vysvetlite, čím sa líšia a čo majú spoločné návrhové vzory Prototype a Factory Method.

Oba slúžia na vytváranie objektov. Factory Method definuje interface na vytváranie objektov ale o tom, ktorá trieda sa vytvorí nechá rozhodovať podtriedy. Prototype vytvára nové objekty kopírovaním prototypu.

48. Stručne (max. 1-2 vetami) vysvetlite, čím sa líšia a čo majú spoločné návrhové vzory Proxy a Adapter.

Adapter poskytuje rozdielny interface pre dany objekt a Proxy poskytuje ten isty interface pre dany objekt. Adapter je urceny pre zmenu interfacu existujuceho objektu. Oba navrhove vzory implementuju objekt pre ovladanie ineho objektu.

49. Stručne (max. 1-2 vetami) vysvetlite, ako súvisia vzory Visitor a Composit.

Poskytnúť operáciu pracujúcu s objektami rôznych typov (tvoriacich zložitejšiu štruktúru) - visitor

Poskytnúť jednotné rozhranie pre prácu so samostatnými objektami aj kontainerom. - composite

Visitor riesi poskytnutie operacie pracujucej s objektami roznych typov a composite poskytuje jednotne rozhranie pre pracu so samotnymi objektami aj kontajnerom

50. Stručne (max. 1-2 vetami) vysvetlite, čím sa líšia a čo majú spoločné návrhové vzory Proxy a Decorator.

Oba vzory obaluju instanciu existujuceho rozhrania (vnutornej instancie), ktore implementuje to iste

rozhranie a deleguje

volania svojich funkcii na rovnake funkcie vo svojej vnutornej instancii.

Dekorator sa zameriava na dynamicke pridavanie funkcii objektu a Proxy na kontrolu pristupu k objektu

51. Stručne (max. 1-2 vetami) vysvetlite, čím sa líšia a čo majú spoločné návrhové vzory Composite a Decorator.

Struktura Composite a Decorator vyzera rovnako len maju rozdielny zamer. Composite dava jednotny interface pre leaf a composite. Decorator pridava funkcie leafu a zaroven jednotny interface.

Vymenujte

52. Vymenujte základné pojmy AOP

Aspekt – Prostriedok modulárneho vyjadrenia pretínajúcich aktivít

Join point – dobre definované miesto v kóde programu komponentu, v ktorom je možné pripojenie kódu rady aspektu

Advice – definuje kód pripájaný v označených bodoch spojenia

Pointcut – vyberá (označuje) body spojenia, ku ktorým je pripojený kód rady

Weaving – program vykonávajúci spojenie (linkovanie) komponentov a aspektov do výsledného kódu

Component – zdrojový kód, do ktorého sú nalinkovane aspekty

53. Vymenujte základné princípy, na ktorých stojí architektúra frameworku Spring.

Inversion of Control - Každý modul sa "sústredí" iba na to, na čo je určený

Dependency Injection - Vytváranie objektov, na ktoré sa spoliehajú iné objekty počas compile-time (automatické vytvorenie a poskytnutie objektu, ktorý pre svoju funkcionalitu potrebuje iný object)

Aspect-Oriented Programming (AOP) - Na oddelenie Cross-Cutting logiky od Business logiky aplikácie

54. Vymenujte typy Dependency Injection, ktoré podporuje Spring

Setter injection, Constructor injection, field injection, lookup method injection

55. Aké 4 hlavné typy noSQL databáz poznáme? (Vymenujte)

Dokumentové databázy (Document databases)

Kľúč-Hodnota databázy (Key-value databases)

StĺpcovO orientované databázy (Column-oriented databases)

Grafové databázy (Graph databases)

56. Aké sú rôzne typy mapovania pre ORM? (Vymenujte)

Mapovanie pomocou XML

Mapovanie pomocou atribútov

Mapovanie kódom

Konvenčné mapovanie (Automapping)

57. Charakterizujte návrhový vzor Observer Vymenujte triedy/objekty tvoriace navrhovy vzor a charakterizujte strucne ich ulohu v nom

Observable - abstraktna trieda

Concrete observable - udrziava stav objektu

Observed - sluzi na pracu s funkcionalitami

concrateObservable A - konkretna implementacia observed

concrateObservable B - konkretna implementacia observed

spark-api

58. Pri ktorých z nasledujúcich operácii spark-api môže nastať premiešanie (shuffle)?

```
coalesce
Cogroup
groupBy
join
repartition
sortByKey
Všetky *ByKey (groupByKey, reduceByKey…) – okrem countByKey
```

59. Ktoré z nasledujúcich operácii spark-api sú transformácie?

```
distinct
Filter
flatMap
flatMapValues
groupBy
intersection
map
mapToPair
reduceByKey
sample
sortByKey
union
```

60. Ktoré z nasledujúcich operácii spark-api sú akcie?

```
aggregate
Collect
count
countByValue
first
fold
foreach
reduce
take(n)
top
```

lambda

61. Ktoré z uvedených lambda výrazov spĺňajú podmienky kladené na argumenty operácie filter?

Predpokladajte, že argumenty a, b sú čísla, s, t sú reťazce

```
S -> “hello”.statWith(s)
a -> a - 3 > 0
s->s.isEmpty()
A -> true
```

62. Ktoré z uvedených lambda výrazov spĺňajú podmienky rýdzej funkcie (pure function)

Môžete predpokladať, že argumenty:
- a,b sú čísla, s reťazec, u je objekt, ktorý ma property urok,
- x je lokálna premenná, y globálna premenná

```
s -> new Tuple2(s, 1)
s -> new Tuple2(a, a*a)
s -> s.isEmpty()
s -> "hello".statWith(s)
a -> { int x=a>0?1:0; return x*a; }
(a,b) -> a + b
a -> Integer.MIN_VALUE + a
s -> { if ("hello".statWith(s)) return true; return false; }
a -> { int x=a>0?1:0; return x*a; }
```

63. Ktoré z uvedených lambda výrazov spĺňajú podmienky kladené na argumenty operácie reduce?

Predpokladajte, že argumenty a, b sú čísla, s, t sú reťazce

hint(Musí to byť komutatívna a asociatívna binárna operácie, vstup viac arg, výstup 1 arg)

```
(a, b) -> Math.min(a, b)
(a, b) -> a && b
(a, b) -> a + b
(a,b) -> a * b
(a, b) -> a || b
(a, b) -> Math.max(a, b)
```

64. Ktoré z uvedených lambda výrazov spĺňajú podmienky kladené na argumenty operácie map?

Predpokladajte, že argumenty a, b sú čísla, s, t sú reťazce

```
A -> new Tuple2(a, a*a)
A -> true
s -> s.isEmpty()
s -> { return s.length(); }
a -> 1.0
s -> s.substring(1, 3)
a -> a > 0
a -> {int y=a>0?1:0; return a*y;}
```

Čo vypíše

65. Čo vypíše nasledujúci program?

```
1. static int LIMIT;
2.
3. public static void main(String[] args) {
4.       SparkConf conf = new SparkConf();
5.       JavaSparkContext sc = new JavaSparkContext(conf);
6.       LIMIT = 20;
7.       List<Integer> dl = Arrays.asList(1, 10, 100, 1000);
8.       JavaRDD<Integer> rdd1 = sc.parallelize(dl);
9.       JavaRDD<Integer> rdd2 = rdd1.filter(x -> x<LIMIT);
10.   LIMIT=200;
11.      rdd2.cache()collect();
12. System.out.println("" + rdd2.count());
13.}
```

3

66. Čo vypíše nasledujúci program?

```
1. static int LIMIT;
2.
3. public static void main(String[] args) {
4.       SparkConf conf = new SparkConf();
5.       JavaSparkContext sc = new JavaSparkContext(conf);
6.       LIMIT = 20;
7.       List<Integer> dl = Arrays.asList(1, 10, 100, 1000);
8.       JavaRDD<Integer> rdd1 = sc.parallelize(dl);
9.       JavaRDD<Integer> rdd2 = rdd1.filter(x -> x<LIMIT);
10.      rdd2.cache()collect();
11.      LIMIT=200;
12.      System.out.println("" + rdd2.count());
13.}
```

2

67. Čo vypíše nasledujúci program?

```
1. static int ZLAVA;
2.
3. public static void main(String[] args) {
4.       SparkConf conf = new SparkConf();
5.       JavaSparkContext sc = new JavaSparkContext(conf);
6.
7.       ZLAVA = 25;
8.       List<Integer> dl = Arrays.asList(100,110,120,130,140,150);
9.       JavaRDD<Integer> rdd1 = sc.parallelize(dl);
10.      JavaRDD<Integer> rdd2 = rdd1.map(x -> x - ZLAVA);
11.      ZLAVA = 5;
12.      JavaRDD<Integer> rdd3 = rdd2.map(x->x-ZLAVA).map(x->x-ZLAVA);
13.      System.out.println("" + rdd3.filter(x -> x < 100).count();
14.}
```

2

68. Čo vypíše nasledujúci program?

```
1. public static void main(String[] args) {
2.       SparkConf conf = new SparkConf();
3.       JavaSparkContext sc = new JavaSparkContext(conf);
4.       LIMIT = 20;
5.       List<Integer> dl = Arrays.asList(1, 10, 100, 1000);
6.       JavaRDD<Integer> rdd1 = sc.parallelize(dl);
7.       JavaRDD<Integer> rdd2 = rdd1.filter(x -> x<LIMIT);
8.       LIMIT=200;
9.       System.out.println("" + rdd2.count());
10.}
```

build error

69. Čo vypíše nasledujúci program?

```
3. public static void main(String[] args) {
4.       SparkConf conf = new SparkConf();
5.       JavaSparkContext sc = new JavaSparkContext(conf);
6.
7.       int ZLAVA = 25;
8.       List<Integer> dl = Arrays.asList(100,110,120,130,140,150);
9.       JavaRDD<Integer> rdd1 = sc.parallelize(dl);
10.      JavaRDD<Integer> rdd2 = rdd1.map(x -> x - ZLAVA);
11.      ZLAVA = 5;
12.      JavaRDD<Integer> rdd3 = rdd2.map(x->x-ZLAVA).map(x->x-ZLAVA);
13.      System.out.println("" + rdd3.filter(x -> x < 100).count();
14.}
```

build error

Predpokladajte

70. Predpokladajte, že sme do kolekcie JavaRDD<String> rdd načítali riadky textového súboru. S využitím operacií RDD-api napíšte výraz, ktorý vráti počet rôznych slov dlhších ako 2 (Pozn. riadky treba rozdeliť na slová)

```
rdd.flatMap(line -> Arrays.asList(line.split(" ")).iterator()).distinct() .filter(word
   -> word.length() > 2) .count();
```

71. Predpokladajte, že máte dve kolekcie JavaRDD<String> rd1 a JavaRDD<String> rd2 obsahujúce reťazce. S využitím operacií RDD-api napíšte výraz pre výpočet symetrickej diferencie množín reťazcov t.j. celkového počtu rôznych reťazcov, ktoré sa nachadzajú práve v jednej z kolekcií (ale nie v oboch)

```
rd1.union(rd2).distinct().subtract(rd1.intersection(rd2)).count();
```

72. Predpokladajte, že máte dve kolekcie JavaRDD<String> rd1 a JavaRDD<String> rd2 obsahujúce reťazce. S využitím operácií RDD-api napíšte výraz pre výpočet symetrickej diferencie množín reťazcov t.j. celkového počtu reťazcov, ktoré sa

nachádzajú len v jednej z kolekcií ale nie v oboch. Reťazce líšiace sa len VEĽKOSŤOU PÍSMEN považujte za totožné.

```
rd1 = rd1.map(s -> s.toUpperCase());

rd2 = rd2.map(s -> s.toUpperCase());

return rd1.union(rd2).distinct().subtract(rd1.intersection(rd2)).count();
```

73. Predpokladajte, že JavaRDD<String> rdd je kolekcia reťazcov. S využitím operacií RDD-api napíšte výraz, ktorého výstupom je boolovská hodnota hovoriaca či sú v kolekcii duplicitné reťazce, t.j. ak sú má výraz hodnotu true inak false.

```
rdd.distinct().count() != rdd.count();
```

74. Predpokladajte, že JavaRDD<String> rdd je kolekcia slov. S využitím operacií RDD-api napíšte výraz, ktorého výstupom je mapa (java.util.map) početností jednotlivých slov v kolekcii (t.j. kľúč je slovo a hodnota je počet výskytov). Slová líšiace sa len veľkosťou písmen považujte za totožné.

```
rdd.map(s -> s.toLowerCase()).countByValue();
```

75. Predpokladajte, že máte dve kolekcie JavaRDD<String> rdd1 a JavaRDD<String> rdd2 obsahujú riadky dvoch textových súborov. S využitím operacií RDD-api napíšte výraz, ktorý vráti zoznam (java.util.List) obsahujúci všetky rôzne slová, ktoré sa nachádzajú v prvom súbore ale nenachádzajú v druhom. (Pozn. riadky treba rozdeliť na slová)

```
rdd1.flatMap(s -> Arrays.asList(s.split(" ")).iterator()).subtract(rdd2.flatMap(s ->
   Arrays.asList(s.split(" ")).iterator())).collect();
```

76. Predpokladajte, že sme do kolekcie JavaRDD<String> rdd načítali riadky textového súboru. S využitím operacií RDD-api napíšte výraz, ktorého výstupom je mapa (java.util.map) početností jednotlivých slov v súbore (t.j. kľúč je slovo a hodnota je počet výskytov). (Pozn. riadky treba rozdeliť na slová)

```
rdd.flatMap(s -> Arrays.asList(s.split(" ")).iterator()).countByValue();
```

77. Predpokladajte, že kolekcie JavaRDD<String> rdd1 a JavaRDD<String> rdd2 obsahujú riadky dvoch textových súborov S využitím operacií RDD-api napíšte výraz, ktorý vráti zoznam (java.util.List) obsahujúci všetky rôzne slová, ktoré sa nachádzajú v prvom súbore ale nenachádzajú v druhom. (Pozn. riadky treba rozdeliť na slová)

```
rdd1.flatMap(s -> Arrays.asList(s.split(" ")).iterator()).subtract(rdd2.flatMap(s ->
   Arrays.asList(s.split(" ")).iterator())).collect();
```

78. Predpokladejte, že máte dve kolekcie JavaRDD<String> rd1 a JavaRDD<String> rd2 obsahujúce reťazce. S využitím operácii RDD-api napíšte výraz pre výpočet symetrickej diferencie množín reťazcov t.j. Celkového počtu reťazcov, ktoré sa nachádzajú len v jednej z kolekcií (ale nie v oboch). Reťazce líšiace sa len prázdnymi znakmi na začiatku a konci reťazca považujte za totožné.

```
rd1 = rd1.map(s -> s.trim());

rd2 = rd2.map(s -> s.trim());

rd1.union(rd2).subtract(rd1.intersection(rd2)).count();
```

79. Predpokladajte, že JavaRDD<String> rdd je kolekcia reťazcov. S využitím operacií RDD-api napíšte výraz, ktorého výstupom je boolovská hodnota hovoriaca či sú v

kolekcii duplicity (t.j. ak sa v kolekcii vyskytuje reťazec viac krát výraz hodnotu true inak false). Reťazce líšiace sa len veľkosťou písmen považujte pri tom za rovnaké.

```
rdd.map(s -> s.toLowerCase()).distinct().count() != rdd.count();
```

80. Predpokladajte, že sme do kolekcie JavaRDD<String> rdd načítali riadky textového súboru. S využitím operacií RDD-api napíšte výraz, vráti počet rôznych slov v súbore, pričom slová líšiace sa len veľkosťou písmen považujte za totožné. (Pozn. riadky treba rozdeliť na slová)

```
rdd.flatMap(s -> Arrays.asList(s.split(" ")).iterator()).map(s ->
   s.toLowerCase()).distinct().count();
```

81. Predpokladajte, že JavaPairRDD<String, List<String>> pdd je kolekcia dvojíc, kde prvá zložka je meno študenta druhá názvov predmetu, ktorý má zapísaný. S využitím operacií RDD-api napíšte výraz, ktorý pre každého študenta vypíše na štandardny výstup riadok obsahujúci meno študenta a reťazec zložený z názvov jeho predmetov oddelených čiarkou. (napr. Fero ASOS,VSA,RZZ)

```
pdd.collectAsMap().forEach((k,v) -> System.out.println(k + " " + String.join(",",
   v)));
```

82. Predpokladajte, že kolekciu JavaPairRDD pdd sme vytvorili načítaním textových súborov funkciou wholeTextFiles S využitím operacií RDD-api napíšte výraz, ktorého výstupom je mapa (java.util.map) udávajuca počet rôznych slov v každom súbore (t.j. kľúč je meno súboru a hodnota počet)

```
Map<String,Long> rddx = rdd.flatMapValues(x-
   >Arrays.asList(x.split("\\s")).iterator()).distinct().countByKey();
```

83. Predpokladajte, že JavaPairRDD pdd je kolekcia dvojíc, kde prvá zložka je meno študenta druhá názvov predmetu, ktorý má zapísaný. S využitím operacií RDD-api napíšte výraz, ktorého návratovou hodnotou je mapa (java.util.map) udávajúca pre každý predmet, koľko študentov ho má zapísaný (t.j. kľúč je názov predmetu a hodnota počet)

```
pdd.values().countByValue();
```

84. Predpokladajte, že sme do kolekcie JavaRDD<String> rdd načítali riadky textového súboru. S využitím operacií RDD-api napíšte výraz, vráti počet výskytov slova ASOS v súbore, pričom nezáleží na veľkosti písmen (Pozn. riadky treba rozdeliť na slová)

```
rdd.flatMap(s -> Arrays.asList(s.split(" ")).iterator()).map(s ->
   s.toLowerCase()).filter(s -> s.contains("asos")).count()
```

85. Predpokladajte, že sme do kolekcie JavaRDD rdd načítali riadky textového súboru. S využitím operacií RDD-api napíšte výraz, vráti dĺžku najdhšieho slova v súbore (Pozn. riadky treba rozdeliť na slová)

```
rdd.flatMap(s -> Arrays.asList(s.split("
   ")).iterator()).map(String::length).reduce(Math::max)
```

86. Predpokladajte, že JavaPairRDD<String, String> pdd je kolekcia dvojíc, kde prvá zložka je meno študenta druhá názvov predmetu, ktorý má zapísaný. S využitím operacií RDD-api napíšte výraz, ktorý pre každého študenta vypíše na štandardny výstup riadok obsahujúci meno študenta a reťazec zložený z názvov jeho predmetov oddelených čiarkou. (napr. Fero ASOS,VSA,RZZ)

```
pdd.flatMapValues(x->Arrays.asList(x.split("
   ")).iterator()).groupByKey().collectAsMap().forEach((k, v) -> System.out.println(k
   + " " + String.join(",", v)));
```

Otázky z prezentácií

87. Čo nepatrí do návrhového vzoru "Factory method"?

Client

88. Majme situáciu, v ktorej do aplikácie prirábame modul na konverziu videí. Chceme použiť komplexnú knižnicu na prácu s videami, pomocou ktorej vieme vytvoriť rýchly a efektívny VideoConverter. Aký štrukturálny návrhový vzor je vhodné v tejto situácií použiť?

Facade

89. Čo nepatrí do štruktúry "Iterator"?

Product

90. Vyberte správne tvrdenia ohľadom Mediator a Observer patternu

Ak komponent urobí nejakú zmenu, Observer upozorní všetky komponenty, ktoré sú na neho napojené.
Mediator rozhodne, ktorý komponent má reagovať na zmenu iného komponentu.
Mediator eliminuje priame prepojenia a závislosti medzi komponentami.
Observer nastavuje dynamické jednosmerné spojenie medzi komponentami.

91. Objekty rodičovskej triedy by sa mali dať nahradiť objektami jej podtried bez toho, aby sa porušila aplikácia. O akom zo SOLID princípov hovorí táto veta?

Liskov Substitution principle

92. Aký problém rieši návrhový vzor Visitor?

Double Dispatch

93. Pri ktorom z príkladov sa môžu antipatterny prejaviť v kóde?

Bloky nepoužitého kódu
Nedostatočná dokumententácia

94. Ktorá z nasledujúcich databáz má grafovú štruktúru?

Neo4j

95. Aké noSQL DB poznáme?

Grafová DB
Dokumentová DB

96. Čo nepatrí k dokumentovým DB?

Redis

97. Aké 4 hlavné typy noSQL databáz poznáme? (Vymenujte)

Dokumentové databázy (Document databases)

Kľúč-Hodnota databázy (Key-value databases)

StĺpcovO orientované databázy (Column-oriented databases)

Grafové databázy (Graph databases)

98. Vyber pojmy tykajuce sa search engines

index
dokument

99. Ktorý vyhľadávací nástroj je IBA cloud based?

Algolia

100. Aké sú rôzne typy mapovania pre ORM?

Mapovanie pomocou XML

Mapovanie pomocou atribútov

Mapovanie kódom

Konvenčné mapovanie (Automapping)

101. V návrhovom vzore MVC figurujú 3 vrstvy. Model, view a controller. Čo zabezpečuje vrstva controller?

Prepája komponenty, spracováva požiadavky používateľov a odovzdáva ich do view na vykreslenie.

102. Je mozne definovat fallback volanie pri pouziti Feign client-a?

Áno

103. Čo sa považuje za výhody pre Spring Data?

Automatické generovanie queries
Repository pattern

104. Ktorá z možností je určená na Spring Messaging?
Spring Integration

105. Ako funguje broker pri protokole AMQP?

Ukladá správy od publishera do fronty aby ich mohol prečítať consumer

106. Vyberte pravdivé tvrdenia o Spark ML?

Spark ML je knižnica kompatibilná s viacerými jazykmi umožňujúca riešiť problémy strojového učenia.
Spark ML slúži na spracovanie veľkých objemov dát.

107. Ktoré metódy nepatria vo frameworku GraphX do partition strategy?

RandomEdgeCut
EdgePartition3D

108. Označte pravdivé tvrdenia.

Integráciou PyTorch s Apache Spark vieme dosiahnuť rýchlejšie trénovanie neurónovej siete s veľkými dátami.
Integrácia Spark s Tensorflow / Pytorch sa používa na distribuované trénovanie.

109. Aký typ architektúry používa Docker?

Klient-Server

110. Ktorý z uvedených typov testovania softvéru nepatrí pod testovanie výkonnosti (performance testing)?

Functional testing

111. Na čo sa používa programovací jazyk R?

analýza dát, štatistika
čistenie dát a zobrazenie grafov

112. Ktoré sú tri základné entrypointy v GraphQL?

Query, Mutation, Subscription

113. Prečo je zakázaný cyklicky import závislostí v jazyku Golang?

Aby bola kompilácia programu rýchlejšia

114. Čo vraví tzv. "Closing channel principle" , teda princíp uzatvárania kanálov v jazyku Golang?

Kanály by sa mali uzatvárať zo zapisovateľa a to len vtedy ak existuje len 1 zapisovateľ

115. V akým prípadoch vie exchange v RabbitMQ správne priradiť prijímateľovi správu ktorú poslal odosielateľ?

Keď sa routing key aj binding key zhodujú
Exchange je typu “Topic”, routing key je foo.bar a binding key je foo.*

116. Kedy sa odporúča použiť microservice architektúra?

Pri vytváraní aplikácií, ktoré sa často menia a dopĺňajú

117. Čo znamená skratka SAML?

Security Assertion Markup Language

118. Ktorý z nasledujúcich servisov nevyžaduje XML konfiguráciu?

Spring Boot

Others

119. Kedy sa odporúča použiť constructor based DI?

ak sa jedná o objekty, ktoré treba vždy zadať (bez nich vytváraný objekt nevie korektne fungovať) Napr. povinne pole, uveďte meno a priezvisko

120. Ktorý protokol gRPC používa?

HTTP 2.0

121. Majme situáciu, v ktorej do aplikácie prirábame modul na konverziu videí. Chceme použiť komplexnú knižnicu na prácu s videami, pomocou ktorej vieme vytvoriť rýchly a efektívny VideoConverter. Aký štrukturálny návrhový vzor je vhodné v tejto situácií použiť?

Facade

122. Na čo slúži Consul K/V Store ?

Na držanie a poskytovanie konfiguračných properties pre aplikácie

123. Čo nepodporuje Cassandra Query Language?

Cudzie kľúče

124. Čo znamená skratka gRPC?

gRPC Remote Procedure Calls

125. Transformácie definícia:

transformujú RDD objekt na iný RDD objekt, vstup a výstup je na workeroch

Narrow transformácia sú: map, FlatMap, Filter…

Wide transformácie sú: Intersection, Distinct…

126. Ako môže funkcia, ktorá je argumentom operácie filter, pracovať s akumulátorom?

môže ho len modifikovať

127. Ako môže funkcia, ktorá je argumentum operácie map, pracovať s broadcast objektom?

môže ho len čítať

128. Ako môže funkcia, ktorá je argumentom operácie foreach, pracovať s akumulátorom?

môže ho len modifikovať

129. Stručne vysvetlite rozdiely medzi návrhovými vzormi Template method a Strategy

Pri strategy sa typicky vytvorí rozhranie(interface) a vykonanie akcie sa deleguje na implementáciu daného rozhrania. Pri template sa využíva dedičnosť a podtrieda override-ne požadované metódy a vykoná v nich potrebné kroky.

130. Stručne vysvetlite rozdiely medzi návrhovými vzormi Adapter a Facade

Adapter sa používa keď wrapper musí použiť určené rozhranie a podporovať polymorfizmus. Používa sa na prepojenie dvoch nekompatibilných rozhraní. Facade sa používa na uľahčenie práce s rozhraním alebo jeho zjednodušenie, napríklad spojenie viacerých malých úkonov do jedného volania.

131. S akým návrhovým vzorom súvisí uzáver (closure) funkcie? Stručne vysvetlite.

Command pattern.

132. Uveďte rôzne type proxy objektov

remote, virtual, protection

133. Uveďte dva rôzne spôsoby implementácie adaptéra

object, class

134. Vysvetlite stručne čo popisuje WSDL element <binding>

Poskytuje konkrétne podrobnosti o tom, ako sa operácia typu portType prenesie.

135. Čo popisuje WSDL element <portType>

Tento element kombinuje viacero elementov na vytvorenie jednosmernej alebo round-trip operacie. Napriklad jeden request a jednu response spravu dokaze skombinovat do jednej request/response operacie.

136. Čo popisuje WSDL element <message>

The messages used by the web service

137. Čo popisuje WSDL element <types>

The data types used by the web service

Na definiciu typov sa pouziva XML-schema jazyk.

138. Charakterizujte návrhový vzor Command

-ukladá requesty do fronty, requesty sa môžu vykonávať v odlišnom čase

Client - nastavi na zaciatku receivera a vytvori Concrete Command

Receiver - aplikuje request

ConcreteCommand - metodou execute vykonava request

Command - interface, ktory cez ConcreteCommand plni request

Invoker - vola jednotlive requesty

139. Aký návrhový vzor sa uplatní pri parsovaní XML-dokumentov? Stručne vysvetlite.

Visitor – podobné operácie sa vykonávajú na objektoch rôznych typov, ktoré spolu tvoria určitú štruktúru. Môžeme vytvoriť samostatnú triedu konkrétnych visitorov, ktorí budú zodpovední za spracovanie rôznych XML elementov.

140. Spark rozdeluje funkcie na pracu s distribuovanymi datami na akcie a transformacie. Vysvetlite v com je hlavny rozdiel medzi nimi.

Transf- transformuju RDD objekt na iny RDD. Vstup a vystup je na workeroch.

Akcie- zobrazuju upravene vystupy, vystup je na driveroch

141. S akými návrhovými vzormi súvisí AOP

proxy, singleton, template method, decorator, observer, command, CoR

142. Uveďte, ktoré návrhové vzory bývajú zvyčajne implementované ako Singleton.

Napr. Abstract factory, Prototype, Builder. Ale napriklad aj Facade sa zvykne implementovat ako singleton lebo jeden facade objekt je vo vacsine pripadov postacujuci.

143. Vysvetlite stručne úlohu IoC kontajnera.

slúži na implementáciu automatického DI. Vytvára požadované objekty, spravuje ich životnosť a automaticky ich injectuje do potrebných objektov.

144. S akým návrhovým vzorom súvisí IoC kontainer frameworku Spring? Stručne vysvetlite.

Template method, Observer a Dependency. Inversion of control sluzi na zmenu flow na zaklade objektoveho grafu, ktory je instancovany assemblerom a dovoluje aby boli objektove interakcie definovane cez abstrakcie

145. Vysvetlite, čo je premiešanie (shuffle) a kedy nastáva, uveďte príklad spark funkcie, pri ktorej môže nastať.

Niektore operacie v Spark spustaju event znamy ako shuffle. Je to event pri ktorom sa redistribuuju data aby boli zoskupene v roznych particiach. Napriklad reparation,sortByKey, join

146. RANDOM KOD – dlzka najkratsieho slova v subore

```
fwdd.mapValues(s ->s.length()).reduceByKey(Math::min).collectAsMap().forEach((k,v)->System.out.println(k + " : " + v));
```

147. RANDOM KOD – pocet roznych slov

```
fwdd.distinct().countByKey().forEach((k,v)->System.out.println(k+ ":" + v));
```