OOP Kvíz 3

05. 03. 2026

Prednáška 3

Celkový počet bodov: 26

* Povinné

* Tento formulár zaznamená vaše meno, zadajte svoje meno.

1

Aký je hlavný vzťah, ktorý modeluje dedičnosť (inheritance) v objektovo orientovanom programovaní? (1 bod)

- ◯ Vzťah "používa" (Uses-a)
- ◯ Vzťah "má" (Has-a)
- ◯ Žiadna z uvedených možností
- ◯ Všetky možnosti sú správne
- ◯ Vzťah "vlastní" (Owns-a)
- ◯ Vzťah "je" (Is-a)
- ◯ Vzťah "vytvára" (Creates-a)

2

Ktoré tvrdenie je správne? (1 bod)

```
interface X {}
class A {}
class B extends A implements X {}
```

- Trieda B môže dediť ešte od ďalšej triedy pomocou extends
- Trieda B implementuje A
- Trieda B dedí od A a implementuje X
- Ak zavoláme new B(); tak najskôr sa vytvorí Interface X a až potom sa začne vytvárať Trieda A a následne Trieda B
- Trieda B dedí od X

3

V akom poradí sa vypíšu nasledujúce reťazce do konzoly?

Volanie v main je:
new B(); (Počet bodov: 2)

```
class A {
    static { System.out.println("A static"); }
    { System.out.println("A instance"); }
    A() { System.out.println("A constructor"); }
}

class B extends A {
    static { System.out.println("B static"); }
    { System.out.println("B instance"); }
    B() { System.out.println("B constructor"); }
}
```

- B instance
- B static
- A constructor
- A instance
- A static
- B constructor

4

Koľko tried môže v Jave priamo dediť jedna trieda (priama dedičnosť)? (1 bod)

- Žiadnu, Java nepodporuje dedičnosť tried
- Maximálne dve
- Neobmedzené množstvo
- Iba jednu (Single Inheritance)

5

Trieda HybridDevice potrebuje kombinovať funkcionalitu tried Printer a Scanner. Ktoré z nasledujúcich riešení je v Jave **správne a rešpektuje pravidlá dedičnosti tried**? (1 bod)

```
class HybridDevice extends Printer implements Scanner { }
```

(za predpokladu, že Scanner je rozhranie)

```
class HybridDevice extends Printer, Scanner { }
```

Možnosť 1

```
class HybridDevice extends Printer extends Scanner { }
```

Možnosť 4

```
class HybridDevice implements Printer, Scanner { }
```

(Printer aj Scanner sú triedy)

6

Čo sa stane, ak v Jave nedefinujete žiadny konštruktor v triede? (1 bod)

- Nemožno vytvárať inštancie tejto triedy
- Trieda sa stane abstraktnou
- Kompilátor vyhodí chybu
- Kompilátor automaticky vytvorí bezparametrický (default) konštruktor

7

Akú funkciu plní volanie super() v konštruktore podtriedy? (1 bod)

- Ukončí vykonávanie konštruktora
- Vytvorí dedenie Triedy
- Vytvorí novú inštanciu aktuálnej triedy
- Zavolá konštruktor priameho dediča (podtriedy)
- Zavolá konštruktor priameho predka (nadtriedy)
- Definuje triedu ako final
- Ohodnotí Triedu že je super
- Zavolá špeciálnu metódu v rámci svojej Triedy, ktorá má rovnaký názov ako daná Trieda,
- Žiadna z uvedených

8

Pozrite si nasledujúci kód. Čo musí byť na mieste otáznikov, ak chceme správne zavolať konštruktor predka?
public class Dog extends Animal { public Dog(String name) { ???(name); } } (1 bod)

- this
- extends
- private
- entends.
- implement
- super.
- this.
- public
- parent
- static
- super

9

Kedy sa vykonáva inicializácia inštančných premenných triedy? (1 bod)

- ◯ Po zavolaní konštruktora predka, ale pred telom vlastného konštruktora
- ◯ Žiadna z uvedených možností
- ◯ Pred zavolaním konštruktora predka (super)
- ◯ Súčasne s vykonávaním statických blokov
- ◯ Až po vykonaní celého konštruktora

10

Čo znamená modifikátor prístupu 'protected'? (1 bod)

- ◯ Člen je prístupný len v rámci podtried
- ◯ Člen je prístupný v rámci balíka (package) a v podtriedach (aj v iných balíkoch)
- ◯ Žiadna z uvedených možností
- ◯ Člen je prístupný odkiaľkoľvek
- ◯ Člen je prístupný len v rámci tej istej triedy
- ◯ Člen je prístupný iba v podtriedach, ale nie v rámci balíka

11

Čo je to "method overriding" (prekrývanie metód)? (1 bod)

- ◯ Skrytie statickej metódy predka
- ◯ Volanie privátnej metódy z inej triedy
- ◯ Všetky možnosti sú správne
- ◯ Vytvorenie metódy s rovnakým názvom ale inými parametrami v tej istej triede
- ◯ Žiadna z uvedených možností
- ◯ Je to polymorfizmus počas behu programu
- ◯ Implementácia metódy v podtriede, ktorá má rovnakú signatúru ako metóda v nadtriede

12

Ktorá anotácia sa odporúča používať pri prekonaní metód pre kontrolu kompilátorom? (1 bod)

- @Inherit
- @Static
- @Extends
- @Overload
- @Polymorphism
- @Super
- @Override

13

Ak použijeme kľúčové slovo 'final' na metódu, čo tým spôsobíme? (1 bod)

- Metóda nemôže byť preťažená (overloaded)
- Metóda musí byť statická
- Metóda nemôže byť prekrytá (overridden) v podtriede. Jediná výnimka je, ak je zároveň aj statická
- Metóda nemôže byť prerytá (overridden) v podtriede
- Žiadna z uvedených možností
- Metóda nemôže byť prekrytá (overridden) v podtriede. Jediná výnimka je, ak je bezparametrická
- Metóda nemôže byť volaná mimo triedy

14

Čo platí pre triedu deklarovanú ako 'final' (public final class X)? (1 bod)

- Jej názov musí byť napísaný veľkými písmenami inak kompilátor vyhodí chybu
- Všetky jej metódy sú automaticky abstraktné
- Nemôže mať žiadny parameter
- Nemôže mať žiadne inštancie
- Nemôže obsahovať žiadne premenné
- Žiadna z uvedených možností

15

Čo je to polymorfizmus v kontexte OOP? (1 bod)

- Schopnosť metódy prijímať iba jeden typ parametra
- Proces kompilácie kódu do bajtkódu
- Žiadna z uvedených
- Vytváranie kópií objektov v pamäti
- Možnosť mať viac konštruktorov v triede
- Automatická konverzia medzi dátovými typmi
- Dynamické viazanie metód – výber implementácie metódy prebieha za behu programu podľa skutočného typu objektu.

16

Ktoré z nasledujúcich tvrdení sú správne - ktorý kód nevypíše chybu? (viac odpovedí) (Počet bodov: 2)

```
class Animal {
    void sound() { System.out.println("Animal"); }
}

class Dog extends Animal {
    void sound() { System.out.println("Dog"); }
    void fetch() { System.out.println("Fetch"); }
}
```

```
Animal a = new Dog();
a.fetch();
```

Možnosť 3

```
Animal a = new Dog();
a.sound();
```

Možnosť 4

```
Dog d = (Dog) new Animal();
```

Možnosť 5

```
Animal a = new Animal();
Dog d = (Dog) a;
```

Možnosť 2

```
Animal a = new Dog();
Dog d = (Dog) a;
d.fetch();
```

Možnosť 1

17

Ktorá trieda je priamym alebo nepriamym predkom všetkých tried v Jave? (1 bod)

- java.lang.Object
- java.lang.toString
- java.lang.Main
- Žiadna z uvedených možností
- java.lang.Static
- java.lang.System
- java.lang.Class

18

Metóda toString() triedy Object štandardne vracia: (1 bod)

- Prázdny reťazec
- To čo napíšeme do konštruktora ako návratovú hodnotu
- Zoznam deklarových metód
- JSON reprezentáciu objektu
- Názov triedy a hash kód objektu (v hexadecimálnom tvare)
- Hodnoty všetkých atribútov objektu

19

Aký je rozdiel medzi operátorom '==' a metódou 'equals()' pri objektoch (ak equals nie je prepísaná)? (1 bod)

- '==' sa používa len pre čísla, 'equals()' pre reťazce
- '==' porovnáva obsah, 'equals()' porovnáva referencie
- Nie je žiadny rozdiel, správajú sa identicky (porovnávajú referencie)
- '==' vždy hádže výnimku pri objektoch

20

Čo znamená princíp substitúcie (Liskov Substitution Principle) v kontexte dedičnosti? (1 bod)

- ◯ Podtriedy musia byť zameniteľné za svoje nadtriedy bez narušenia správnosti programu
- ◯ Metódy by mali mať maximálne 3 parametre
- ◯ Každá trieda musí mať práve jednu inštanciu
- ◯ Dedičnosť by sa nemala používať vôbec

21

Čo sa stane, ak v konštruktore podtriedy zabudnete explicitne zavolať super() a predok nemá bezparametrický konštruktor? (1 bod)

- ◯ Program zlyhá pri spustení (Runtime Exception)
- ◯ Kompilátor zahlási chybu (Compile-time error)
- ◯ Vytvorí sa inštancia bez inicializácie predka
- ◯ Nič sa nestane, lebo Java zavolá implicitný konštruktor
- ◯ Java automaticky doplní volanie super(null)
- ◯ Žiadna z uvedených možností

22

Aký je výsledok nasledujúceho kódu? Object a = "Hello"; if (a instanceof String) { System.out.println("Je to String"); } (1 bod)

- ◯ Vyhodí ClassCastException
- ◯ Nevypíše nič
- ◯ Vypíše "Je to String"
- ◯ Kompilátorová chyba

23

Čo je to "Dynamic Binding" (Dynamická väzba)? (1 bod)

- ◯ Rozhodnutie o tom, ktorá metóda sa vykoná, sa deje až za behu programu (runtime) na základe typu objektu
- ◯ Spojenie metódy s jej implementáciou počas kompilácie
- ◯ Priradenie hodnoty do premennej typu final
- ◯ Statické importovanie knižníc

24

Prečo sa v Jave preferuje kompozícia pred dedičnosťou? (1 bod)

- ◯ Kompozícia poskytuje väčšiu flexibilitu a znižuje pevnú previazanosť (coupling) tried
- ◯ Java nepodporuje dedičnosť
- ◯ Dedičnosť nefunguje s rozhraniami
- ◯ Dedičnosť je pomalšia

25

V akom formáte sa odovzdáva dokument v rámci prvého zadania? *

Napíšte len skratku formátu ako koncovku