Závěrečný test 4IT101: Odpovědi mohou být správně i všechny nebo žádná

Co reprezentují instance třídy java.io.File?

Abstraktní cestu k souboru či složce

Data uložená v souborech, které jsou uloženy ve vnější paměti (disk, flash ...) a uspořádány do stromové struktury

Data uložená na disku

Identifikátor v Javě může obsahovat:

podtržítko _

ampersand &

mezeru

číslice

zavináč @

Identifikátor emanuel obsahuje odkaz na instanci třídy Motýl, která implementuje rozhraní ObyvatelLouky. Které z následujících podmínek vrátí hodnotu true?

- emanuel instanceof Motyl

- emanuel instanceof Serializable

- emanuel instanceof Object

- emanuel instanceof Cloneable

- emanuel instanceof ObyvatelLouky

Jak dělíme datové proudy

vstupní x výstupní

datové proudy nijak nedělíme

bajtové x znakové

otevřené x zavřené

soubory x složky (Datové proudy pro Otevření a uzavření souboru v C a C++)

Jako typ návratové hodnoty metody může být uvedeno:

pole prvků primitivních datových typů

pole prvků objektových (referenčních) typů

jakýkoli objektový (referenční) typ

typ void

Je dána deklarace proměnných int cislo; int desítky;

Který z následujících kódů získá do proměnné desítky číslici z pozice desítek v proměnné cislo ?

(Například aby pro cislo=1234 bylo desitky=3.)

desitky = (cislo % 100)/10

desitky = (cislo/ 100)*10

desitky = (cislo/10)%10

desitky = cislo/10

K čemu se v Javě používá identifikátor:

- Pro pojmenování metody
- Pro pojmenování klíčových slov
- Pro pojmenování proměnné
- Pro pojmenování balíčku

Kde se v javě používají složené závorky?

- pro ohraničení deklarace třídy nebo rozhraní
- jako operátor přetypování
- ve výrazech pro označení priority operací
- při volání metody při uvádění skutečných parametrů metody
- v deklaraci metody pro ohraničení seznamu formálních parametrů
- při přístupu k prvku vícerozměrného pole
- v příkazech selekce a iterace pro uvedení podmínek

Kde se v Javě používají kulaté závorky?

- v deklaraci metody pro ohraničení seznamu formálních parametrů
- při volání metody při uvádění skutečných parametrů metody
- pro ohraničení bloku příkazů

- při určení pořadí položky uvnitř pole

Která tvrzení o seznamech (třídách implementujících rozhraní List<E>) jsou pravdivá a která nepravdivá?

- Seznam může obsahovat prvky primitivních typů

- Seznamy udržují pořadí prvků a je možné používat indexy.

- Seznamy mohou obsahovat libovolný počet shodných prvků

- Seznamy není možné procházet pomocí klasického cyklu for.

- Všechny prvky v seznamu jsou stejného typu nebo jeho podtypů.

- Pomocí metody add s jedním parametrem vkládáme prvek na konec seznamu.

- Indexy v seznamu jsou číselné a začínají od 1.

- V seznamech jsou jako indexy používány řetězce.

Která z uvedených pravidel musí platit pro implementaci metody equals()?

- Musí být tranzitivní: pokud x.equals(y) vrátí true a y.equals(z) také true pakmusí x.equals(z) vrátit true.

- Musí být symetrická: pokud x.equals(y) vrátí true, musí y.equals(x) vrátit true.

- Pro x, které není null, musí x.equals(null) vrátit false

- Pro x, které je rovné null, musí x.equals(null) vrátit true.

Která z následujících pravidel musí platit pro implementaci metody hashCode()?

- Pokud zavoláte metodu hashCode() několikrát za sebou pro tutéž instanci, musí se vždy vrátit stejný výsledek.

- Pokud jsou si dvě instance rovny (metoda equals() při jejich porovnání vrátí true), musí metoda hashCode pro obě instance vrátit stejné číslo.

- Pokud metoda hashCode() pro dvě instance vrátí stejné číslo, znamená to, že jsou tyto instance shodné (metoda equals() při jejich porovnání vrátí true).

- Pokud jsou dvě instance shodné (metoda equals() vrátí true), musí metoda hashCode() pro tyto instance vrátit hodnotu 0.

- Pro dvě rozdílné instance (metoda equals() při jejich porovnání vrátí false) nesmí metoda hashCode() vrátit stejnou hodnotu.

Které modifikátory přístupu lze použít u datových atributů?

- Private
- protected
- public
- modifikátor přístupu neuveden

Které modifikátory přístupu je možné použít u metod instancí?

- Private
- protected
- public
- modifikátor přístupu neuveden

Které z následujících / uvedených metod jsou definované ve třídě Object?

- toString()
- clear()
- hashCode()
- getName()
- equals
- size()
- get Class()
- finalize()
- clone()

Které z následujících cyklů se provedou právě 6× (Šestkrát)?

- for (int i=1; i<12; i=i+2)
- for (int i=-2; i<=10; i=i+2)
- for (int i=-5; i<=5; i=i+2)
- for (int i=6; i>=0; i--)
- for (int i=7; i>-11; i=i-3)
- for (int i=0; i<=6; i++)
- for (int i=1; i<=6; i++)
- for (int i=0; i<6; i++)

Který z následujících identifikátorů je platný v JAVĚ?

- MOJE_KONSTANTA
- 5prstu
- celeCislo
- this
- Něco
- MojeŠikovnáMetoda

Máme dva textové řetězce:String s1="Praha"; String s2="praha"; Které vrátí hodnotu True ?

- s1.length() == s2.length()
- s1.length() != s2.length()
- s1.toUpperCase().equals (s2.toUpperCase();
- s1.equals(s2)

Při kterých použití ve stejné třídě vznikne chyba (při překladu nebo za běhu programu)://předpokládám, že metoda vrací float nebo double

- vratCislo();
- int cislo = vratCislo();
- String cislo = vratCislo();
- double cislo = vratCislo();

Mezi základní objektové vlastnosti patří:

- dědičnost
- možnost definování tříd objektů
- komunikace objektů (posílání zpráv, volání metod)
- existence objektů (instancí)
- zapouzdření a ukrývání implementace

Modifikátor final může být uveden:

- V záhlaví třídy
- v záhlaví konstruktoru
- v záhlaví metody
- U parametru metody
- u formálního parametru metody

Nekonečný cyklus while (true) { ..... } lze uvnitř bloku (cyklu) ukončit:

- vyvoláním výjimky pomocí
- příkazem continue
- příkazem break
- pomocí příkazu goto
- příkazem return
- zavoláním metody System.exit(0)

Označte dvojice hlaviček metod, ve kterých se jedná o přetížení metod v rámci jedné třídy:

- void metoda ( ) { ... a void metoda (int pocet) { ...
- void metoda (double stranaA) { ... a void metoda (double stranaB) { ...
- void metodaA (double stranaA) { ... a metodaA (double stranaB) { ...
- void metodaA ( ) { ... a void metodaB ( ) { ...

Označte části počítače, které jsou součástí historického von Neumannova schématu počítače:

- Vstupně-výstupní zařízení
- Komunikační zařízení
- Řídící obvody (ŘADIČ)
- Monitor
- Paměť
- Pevné disky

Označte pravdivá a nepravdivá tvrzení o konstruktorech:

- Při psaní konstruktoru potomka lze volat předka pomocí super(..)
- Při vytváření instance se provádí konstruktor,ostatní metody pouze pokud jsouvolányz konstruktoru

- V hlavičce konstruktoru musí být uvedeno jméno třídy

- Konstruktory nelze přetěžovat

- Konstruktor nelze volat ze statických metod

- Provádění konstruktoru nelze ukončit pomocí příkazu return

Označte pravdivá tvrzení o debugeru v Javě:

- Zarážky se používají pro vyznačení místa, kde se provádění kódu přerušit.

- Krokování kódu označuje činnost, kdy programátor v debugeru sledujeprovádění kódu po jednotlivých řádcích - stisknutím tlačítka volí okamžik,kdy se má provést další řádek.

- Při zastavení provádění kódu lze vidět v debugeru hierarchie volání metod.

- Pokud se zastaví provádění kódu, lze v debugeru zobrazit obsah datových atributů a lokálních proměnných.

- Pomocí krokování kódu v debugeru se nejčastěji testuje veřejné rozhraní (API) třídy.

- Při krokování kódu lze v debugeru změnit průběh provádění kódu (např. přeskočit některé příkazy).

Označte ty hodnoty proměnné x, pro které je splněna následující podmínka:

(x>2 & x<10) || (x>5 & x <15)

- 6

- 15

- 5

- 3

- 9

- 11

Označte ty hodnoty proměnné x, pro které je splněna následující podmínka:

(x>0 & x<6) || (x>6 & x<10)

- 0

- 1

- 6

- 5

- 10

- 7

Označte ty hodnoty proměnné x, pro které je splněna následující podmínka:

(X<5 || x>15) & (x>10 || x<6)

- -5
- 15
- 0
- 16
- 6

Označte případy, kdy vznikne výjimka:

- int cislo=5 int vysledek = cislo / 0
- int cislo=5 double vysledek = cislo / 0
- double cislo = 5 double vysledek = cislo / 0
- double cislo = 5.0 double vysledek = cislo / 0.0

Přetížené metody se od sebe mohou lišit:

- Počtem parametrů
- jménem metody
- typem a pořadím parametrů
- návratovou hodnotou

Při deklaraci formálních parametrů metody lze uvést:

- modifikátor final
- přiřadit defaultní (implicitní) hodnotu
- určit typ parametru
- modifikátor public

Při zpracování cyklu (kde z je proměnná typu int)

- while(z>=0) , suma+=z mohou nastat tyto případy:
- cyklus neproběhne ani jednou
- Cyklus proběhne právě jednou

- program se zacyklí

- Skončí až bude proměnná z rovna 0

Při zpracování vstupně-výstupních operací mohou být vyhozeny výjimky, jejichž typy mají následující hierarchii: Exception - IOException - FileNotFoundException ... záleží na formulaci otázky ... pozor ...

- Každá metoda, která volá metodu mmm musí použít konstrukci try-catchzachytávající a ošetřující výjimku FileNotFoundException.

- Metoda mmm musí deklarovat, že vyhazuje výjimku IOException neboException.

- Metoda mmm musí deklarovat, že vyhazuje výjimku FileNotFoundException.

- Na uvedené metody nejsou kladeny žádné speciální požadavky.

Shodně se mohou jmenovat:

- dvě metody se stejnou návratovou hodnotou a různými (stejnými) parametry

- dvě metody s různými návratovými hodnotami a různými (stejnými)parametry

Shodně se mohou jmenovat:

- lokální proměnná a datový atribut instance

- formální parametr metody a datový atribut instance

- formální parametr metody a lokální proměnná

- datový atribut instance a metoda instance

Slovo super se používá při:

- volání metody předka

- použití datového atributu předka (pokud k němu má potomek přístup)

- volání konstruktoru předka

- jako odkaz na jinou třídu z téhož balíčku

Uvnitř metody může být deklarace:

- lokální proměnné

- datového atributu instance

- jiné metody

- statické proměnné třídy

Uvnitř těla metody může být:

- příkaz return
- prázdný odkaz
- prázdný příkaz,
- příkazy skoků a cyklu
- deklarace atributu instance

U kterých z následujících zápisů ohlásí překladač chybu?

- Předchází deklarace float abc = 5
- if (abc=5.37), <4.3, (int) 4- , < !=4L,

U kterého z následujících zápisů podmínky příkazu if překladač ohlásí chybu?Kódu předchází deklarace double abc = 5.3;

- if (abc < (int)4)
- if (abc < 4)
- if (abc = 5.37)
- if (abc >= 5.3)

Uvnitř jedné třídy se mohou shodně jmenovat:

- lokální proměnná a datový atribut instance
- formální parametr metody a datový atribut instance
- datový atribut instance a metoda instance
- formální parametr metody a lokální proměnná metody

V Javě může identifikátor začínat:

- Podtržítkem
- Mezerou
- Písmenem
- Zavináčem@

- Ampersandem &

- Číslicí

Vyberte pravdivá tvrzení o datovém typu pointer:

- Proměnná typu pointer může odkazovat na hodnotu, které již byla zrušena.

- Po nevhodné aritmetice s proměnnou typu pointer může proměnná ukazovat jinam, než kam bylo zamýšleno.

- Typ pointer podporuje operace přiřazení adresy, přičítání celého čísla k adrese a odečítání celého čísla od adresy.

- Java podporuje datový typ pointer.

Vyberte pravdivá tvrzení o objektech a abstraktních datových typech

- Třídy v objektových programovacích jazycích jsou příklady abstraktníhodatového typu.

- Abstraktní datový typ (ADT) umožňuje ukrývat implementaci.

- V programu lze deklarovat a inicializovat proměnné abstraktního datového typu.

- Při deklaraci abstraktního datového typu (ADT) může programátor definovatoperace(metody, funkce) pro tento typ.

- Při deklaraci abstraktního datového typu lze použít dědičnost a to i v případě, že se nejedná o třídu v OOP jazycích.

Vyberte pravdivá tvrzení o jednotkovém testování pomocí Junit

- Dle konvencí by jméno testovací třídy mělo tvořit jméno testové třídy a slovo Test

- Pro porovnání očekávané hodnoty se skutečnou návratovou hodnotou se v testechnejčastěji používá metoda assertEquals.

- S pomocí Junit se většinou testuje veřejné rozhraní (API) třídy

- Dle konvencí by jméno testovací metody mělo začínat slovem test

- Pomocí Junit testů lze přímo otestovat privátní metody testované třídy

- Pomocí JUnit testů nelze testovat vznik výjimek v testované metodě.

Vyberte pravdivá tvrzení o vedlejších efektech metody (výrazu):

- Vedlejší efekt metody (výrazu) označuje situaci, kdy metoda (či výraz) mění i jinýstav (proměnnou) procesu, než je návratová hodnota.

- Metody bez vedlejších efektů jsou čitelnější, neboť při pochopení významu senemusí brát v úvahu další proměnné mimo metodu.

- Operátor ++ v Javě je příkladem operátoru s vedlejším efektem - vrací návratovou hodnotu a současně zvyšuje hodnotu příslušné proměnné.

- Metody bez vedlejších efektů deklarované uvnitř třídy nemění hodnotu datovýchatributů instance této třídy.

- Pokud by metody met1 a met2 v následující ukázce byly bez vedlejších efektů,tak nezávisí na pořadí volání následujících dvou příkazů: int a = met1(prom1);int b =met2(prom2);

Vyberte pravdivé výroky o jazyce Prolog:

- Databáze v pojetí Prologu je seznam fakt a pravidel

- Fakta v Prologu popisují vlastnosti objektů a vztahy mezi objekty

- Pravidla umožňují ze stávajících fakt odvozovat další fakta

- Prolog podporuje cykly i v rekurzi

- V Prologu jsou všechny proměnné stejného typu

Vyberte správnou verzi hlavičky metody, která se musí ve třídě nadeklarovat pro spuštění

- Public static void main (String [ ] cokoliv)

- Java aplikace z příkazové řádky:

- Static void main (String [ ] ars)

- Public static void start (String [ ] arametry)

- Public static void main (String args)

Výjimky dělíme na:

- Kontrolované x nekontrolované

- nepoužíváme žádné z uvedených dělení

- aplikační x programové x systémové

- ošetřené x neošetřené

Z konstruktoru lze volat

- jiný konstruktor téže třídy
- konstruktor potomka
- konstruktor předka
- statickou metodu téže třídy

Z následujícího seznamu vyberte funkcionální programovací jazyky:

- C
- LISP
- Perl
- Haskel
- Scheme
- Common Lisp
- Java
- Ruby
- ML

Z následujícího seznamu vyberte programovací jazyky, které byly od začátku navrženy s objekty:

- Cobol
- C#
- C
- LISP
- Prolog
- Java
- Python
- Smalltalk

Záhlaví konstruktoru může obsahovat:

- modifikátor protected
- klauzuli implements
- modifikátor prIvate
- Modifikátor final

- klauzuli throws

- Deklaraci formálních parametrů metody

Záhlaví metody může obsahovat:

- modifikátor protected

- klauzuli implements

- návratovou hodnotu

- klauzuli throws

Záhlaví třídy (samostatně ne vnitřním nebo vnořené) může obsahovat:

- Klauzuli implements

- Klauzuli throws

- Modifikátor Protected

- Návratovou hodnotu

Přiřaďte dokumentační značky pro javadoc:

- k metodám

- k třídám

Značky:

@version @return @param @author @exception

Přiřaďte Java jmenné konvence k jednotlivým identifikátorům

konvence:

VNM, VNV, všechna velká

identifikátory:

- lokální proměnné

- statické konstanty

- metody

- statické proměnné

- třídy

- výčtového typu enum

- rozhraní

Přiřaďte jednotlivé činnosti k lexikální, syntaktické a sémantické analýze

- Odstranění komentářů ze zdrojového kódu probíhá při
- Identifikace klíčových slov (např. if či for v Javě) probíhá při
- Syntaktický strom se vytváří při
- Generování mezikódu je součástí
- Kontrola správného zápisu příkazu při

Přiřaďte k metodám ze třídy Object jednoduchý popis:

Metody:

finalize() ; clone() ; getClass() ; equals() ; hashCode

Popis:

- Metoda sloužící k porovnání instancí (většinou obsahů instancí) **equals()**
- Metoda sloužící pro vytváření kopií instance **clone()**
- Metoda, která vrací číselný kód instance, který se používá např. Při vkládání HashSetu **hashCode**
- Metoda, která vrací informace o třídě **getClass(**
- Metoda, jterou spouští Garbage Collector před odstraněním instance z paměti **finalize()**

Přiřaďte k situaci odpovídající klíčové slovo:

klíčové slovo:

this, Super, PNM, ...

Situace:

- Za statické třídy se můžeme **PNM**
- Z konstruktoru můžeme **Super**
- Pro zavolání konstruktoru **this**
- Pro identifikaci dat.atr. **this**
- Na statickou konstaktu**...**

Přiřaďte k situaci odpovídající klíčové slovo super či this:

- Pro identifikaci datového atributu ve stejné instanci použijeme klíčové slovo **this**

- Pro zavolání konstru. z jiného konstruktoru stejné třídy použijeme klíčové slovo **this**

- Z konstruktoru můžeme zavolat konstruktor předka s pomocí klíčového slova **super**

- Na statickou konstantu(static final)ve st. tř. se můžeme odkázat s pomocí klíčového slova **this**

- Ze statické metody třídy se můžeme na privátní metody instance stejné třídy odkázat pomocí klíčového slova.

Přiřaďte správné koncovky k souborům:

Koncovky:

.class; .ctxt; .enum; .java

Soubory:

bytecode rozraní-**class**

zdroj. kod rozh.-**java**

bytecode (přeložený kod) výčt. typu-**class**

bytecode (přeložený kod) třídy)-**class**

zdroj. kod třídy-**java**

zdroj. kod výčtového typu-**java**

Přiřaďte správné jméno metody ze třídy Math k jejímu popisu:

Metody sqrt; floor, abs, exp, log10, pow, log

Popis:

- absolutní hodnota **abs**

- zaokrouhlení na nižší **floor**

- mocnina **pow**

- přirozený logaritmus **log**

- přir.mocnina **exp**

- druhá odmocnina **sqrt**

- desítkový logaritmus **log10**

Přiřaďte statické proměnné třídy System z balíčku java.lang k odpovídajícímu popisu

Proměnné:

PrintStream System.err; InputStream System.in; PrintStream System.out

Popis:

- vstup z konzoly **InputStream System.in**

- chybový výstup **PrintStream System.err**

- výstup na konzolu **PrintStream System.out**

Přiřaďte způsob zápisu výrazu (infixový, prefixový, postfixový) k jednotlivým příkladům výpočtu obvodu obdélníka:

* 2 + a b **prefixový**

* + a b 2 **prefixový**

(a + b) * 2 **infixový**

2 a b + * **postfixový**

a b + 2 * **postfixový**

2 * (a + b) **infixový**

Rozhodujete se, zda pro evidenci předmětů a známek použít textové soubory či databázi.

- lze definovat přístupová práva k jednotlivým záznamům či položkám **databázi**

- snadná přenositelnost dat **soubory**

- jednodušší používání - není potřeba instalovat, spravovat a učit se další SW **soubory**

- aktualizovat data může současně více uživatelů **databázi**

- udržuje se i popis struktury dat **databázi**

Seřaďte následující řádky tak, jak by měly za sebou následovat v programu

(mezi uvedenými řádky mohou být další řádky programu) (očíslovat od 1 do 5)

try {

} catch(RuntimeException rte) {

} catch(Exception rte) {

} catch(Throwable rte) {

} finally {

Co vypíše následující kód?

int cislo = 24, system.out.println (cislo/5)

4

Co vypíše následující kód?

int cislo = 12, system.out.println (12%9)

3

Co vypíše následující kód :

int cislo=12;/if((cislo%6)==0{/System.out.println("ahoj");}else {System.out.prinln ("nashledanou");}

ahoj

Co vypíše následující kód :

int cislo=15;/if((cislo%6)==0{/System.out.println("ahoj");}else {System.out.prinln ("nashledanou");}

nashledanou

Doplňte v následující metodě podmínku příkazu if, pomocí které zjistíte, zda parametry metody obsahují stejnou hodnotu (nevkládejte mezery):

private void metoda (String s1, String s2) {

if () { // .... pokračování metody (má se doplnit do mezery)

s1.equals(s2)

Hodnota konstanty Math.PI je 3.141592653589793

Co vytiskne následující řádek kódu: System.out.printf ("%5.3f", Math.PI);

3.142

Kterým klíčovým slovem uvozujeme v hlavičce metody seznam vyhazovaných výjimek

Throws

String s1, s2 - s1 = "cacao" - s2=s1.replace("c","k"). Jaká je hodnota proměnné s2?

Kakao

String s1, s2; s1 = " 359 "; s2 = s1.trim(); Jaká je hodnota proměnné s2

359 (metoda trim řetězec zbaví mezer na začátku a na konci)

String s = „mala a VELKA" int i=s.indexOf(´a´) Jakou má hodnotu proměnná i ?

1

V následujícím kódu doplňte část pro zjištění délky řetězce s:String s="mala a VELKA";int delka= s.length();

V následující ukázce se rozdělí textový řetězec na více částí:

String adr = "Karel Novák,Dlouhá 35,Praha 1,101 00";

String [] casti = adr.split(",");

Co vypíše následující řádek kódu: System.out.print (casti[2]);

Praha 1 // cislovani od 0

Co je to pragmatika informace?

Pragmatika informace označuje "praktický význam" zprávy pro osobu příjemce.

Pragmatika informace označuje strukturu informace. Pragmatika se skládá z morfologie a syntaxe.

Pragmatika označuje nezrušitelnou, neměnnou část informace.

Pragmatika informace zkoumá vztah informace k příjemci

Pragmatika informace se používá pro vyjádření takového významu informace, který je nezávislý na příjemci.

Pragmatika informace se používá pro vyjádření takového významu informace, který je nezávislý na příjemci

Pragmatika informace je ta část informace, kterou lze prakticky (obvykle písemně) zachytit

Deklarujete metodu, která má jako parametr pole řetězců: public void metoda(String [ ] pole)

for (int i= 1; i++ ; i<= pole.length) {System.out.println(pole[i]);}

+for (int i= 0; i< pole.length; i++) {System.out.println(pole[i]);

for (int i= 1; i<= pole.length; i++) {System.out.println(pole[i]);}

+for (String polozka : pole) {System.out.println(polozka);}

for (int i= 0; i< pole.length; i+1) {System.out.println(pole[i]);

for (int i=50 i>0 i=i-2). Kolikrát cyklus proběhne?

25x popř. Žádná z předchozích

Cyklus neproběhne ani jednou

Vznikne nekonečný cyklus

24x

Jakou hodnotu vrátí metoda size() volaná v následujícím kódu:

Set string mnozina = new HashSet String ()

add.pes

add.kočka

add.pes

size = 2

Jakou hodnotu vrátí metoda size() volaná v následujícím kódu:

Set string = new HashSet ()

add.pes

add.kočka

remove myš

size= nic -> chyba

Jakou hodnotu vrátí metoda size() volaná v následujícím kódu:

List (String) seznam = new ArrayList(string)

add.pes

add.kocka

get 1

size = 2

Jakou hodnotu vrátí metoda size() volaná v následujícím kódu:

List (String) seznam = new ArrayList(string)

add.pes

add.kocka

get 2

size = vznikne výjimka

Jakou hodnotu vrátí metoda size() volaná v následujícím kódu:

List (String) seznam = new ArrayList(string)

add.pes

add.kocka

addpes1

size = 3

Jakou hodnotu vrátí metoda size() volaná v následujícím kódu:

Map<String,String> seznam = new HashMap<String>();

put("pes","Alík");

put("kočka","Micka");

put("pes", "Rek");

size();

- 2

Mějme dán kód: final int cislo; cislo =5

- Vznikne chyba (výjimka) za běhu programu
- nevznikne žádná chyba, proměnné bude 0
- nevznikne žádná chyba, proměnná bude 5
- Vznikne chyba při překladu

Mějme následující kód:

if (a>4) {

System.out.println("A");}

else {

if (a>9) {

System.out.println("B");}

else {

System.out.println("C");}}

Pro jakou hodnotu proměnné a bude vytištěno B ?

a) pro a menší než 4

b) pro a mezi 4 a 9

c) nikdy

d) pro a větší než 9

e) pro a menší než 0

Můžete v Javě napsat a přeložit třídu, která má v hlavičce uvedenou implementaci rozhraní a při tom neimplementje všechny metody tohoto rozhraní?

Ano může, může to být libovolná třída.

Ano, ale třída musí být označena jako abstraktní.

Ano, ale taková třída musí být označená jako konečná (s modifikátorem final).

Ne, takovou třídu není možné přeložit.

Kolikrát proběhne následující cyklus:

for (int i = 20; i > 1; i--) {System.out.println(i);}

19x

Kolikrát proběhne následující cyklus:

int n = 20; while (n > 0) {System.out.println(n); n++;}

- 19x

- 20x

- 21x

- Ani jednou

Které z následujících prvků mohou být součástí hlavičky metody:

- modifikátor final

- modifikátory vstupu

- typ návratové hodnoty

- příkazy

//hlavička neobsahuje modifikátory vstupu, ale modifikátory přístupu ano

Které z následujících prvků mohou být součástí hlavičky metody

- příkazy

- hodnota null

- typ návratového hodnoty

- klíčové slovo return

Který z následujících fragmentů kódu nám při následném odchycení výjimky poskytne informaci o jméně souboru,

který nebyl nalezen (předpokládejte, že všechny proměnné jsou korektně deklarovány a použity):

- If (!f.exists()) { throw new IOException („Soubor „ + f.getName()+"nenalezen");}

- Exception e=new IOException („Soubor nenalezen"); if (!f.exists()){throw e;}

- if (!f.exists()) { throw „Soubor nenalezen";}

- If (!f.exists()) { throw new IOException();}

Označte pravdivá a nepravdivá tvrzení o balíčcích (packages) v Javě.

- Třídy umístěné v jednom balíčku mohou vzájemně přistupovat k datovým

- atributům a metodám označeným modifikátorem přístupu private.

- Balíčky slouží k vytváření jmenných prostorů, tj. v různých balíčcích mohou být třídy se stejným jménem.

- Balíčky se používají k seskupování tříd pro posílání zpráv, tj. lze poslat jednu zprávu všem třídám v balíčku.

- V jednom balíčku může být několik tříd stejného jména.

Pro identifikátor statické proměnné se v Javě používá následující jmenná konvence:

- Všechna slova jsou velkým písmenem, jednotlivá slova jsou oddělena tečkou. např.:static double BANKOVNI.UROK = 4,5;

- Začátek každého slova je velkým písmenem, např.: static double BankovniUrok= 4,5;

- Všechna slova jsou velkým písmenem, jednotlivá slova jsou oddělena podtržítkem. např.: static double BANKOVNI_UROK = 4,5;

- Začátek každého slova je velkým písmenem s výjimkou prvního slova, které začíná malým písmenem např.: static double bankovniUrok = 4,5;

Přátelský přístup k metodě, atributu, třídě atd. se označuje:

- tím, že v deklaraci neuvedeme modifikátor přístupu

- modifikátorem private

- modifikátorem public

- modifikátorem protected

Při spuštění Java aplikace z příkazové řádky končí obvykle program provedením všech příkazů v metodě public static void main(String [] args).

Jak ukončíte aplikaci uprostřed metody main (nepředpokládáme spouštění vláken z metody main)?

- příkazem break

- příkazem continue

- příkazem return

- aplikaci nelze ukončit uprostřed metody main

- System.exit(x)

Stav instance se uchovává:

- V parametrech metod

- V lokálních proměnných

- V metodách

- V datových atributech

Určete jakou hodnotu nabude proměnná kk:

pro int kk =0 po proběhnutí cyklu: int i=10, while (i>kk, kk++, i--)

- 6

- 4

- 5

- 10

- nekonečno

Určete jakou hodnotu nabude proměnná kk:

pro int kk =0 po proběhnutí cyklu: int i=5, while (i>kk, kk++, i--)

- 5

- 2

- 6

- 3

V metodě deklaruji lokální pole typu int a naplním ho pomocí následujícího cyklu.

Jaké hodnoty budou v poli?

int [] pole = new int [10] ; for (int i=1; i<= pole.length; i++) {pole [i]=i ; }

pole [i] = i;

nepřeloží se

V metodě deklaruji lokální pole typu int a naplním ho pomocí následujícího cyklu.

Jaké hodnoty budou v poli?

int [] pole = new int [10] ; for (int i=0; i<= pole.length; i++) {pole [i]=i ; }

pole [i] = i;

nepřeloží se

V metodě deklaruji lokální pole typu int a naplním ho pomocí následujícího cyklu.

Jaké hodnoty budou v poli?

int [ ] pole = new int[10];

for (int i = 0; i< pole.length; i++) {

pole [i] = i+1;

1 2 3 4 5 6 7 8 9 10

V metodě deklaruji lokální pole typu int a naplním ho pomocí následujícího cyklu.

Jaké hodnoty budou v poli?

int [ ] pole = new int[10];

for (int i = 1; i< pole.length; i++) {

pole [i] = i+1;

2 3 4 5 6 7 8 9 10

V programu potřebujeme zapsat data do souboru, soubor bude připraven pro zápis, když:

- Vytvoříme instanci třídy File a zavoláme metodu open().

- Vytvoříme instanci třídy FileOutputStream nebo FileWriter, jako parametr zadáme soubor a při vytváření instance nevznikne výjimka.

- Pro načtení souboru využijeme metodu System.in.readLine().

- Vytvoříme instanci třídy FileInputStream nebo FileReader, jako parametr zadáme soubor a při vytváření instance nevznikne výjimka.

Vyberte jeden nepravdivý výrok o statických metodách:

- ve třídě A je deklarována statická metoda xyz() a metoda instance abc(). Ze statické metody je možné zavolat metodu instance abc() jednoduchým způsobem: abc()

- Volání statické metody xyz() ze třídy A lze zapsat pomocí odkazu na jméno třídy:A.xyz()

- Ve třídě A je deklarována statická metoda xyz() a metoda istance abc(). Z metody instance je možné zavolat statickou metodu jednoduchým zápisem xyz()

- Statická metoda musí mít uveden v hlavičce modifikátor static

Vyberte pravdivý výrok o deklaraci parametrů metody (tj. co lze uvést u parametrů v hlavičce metody)

- U parametru metody mohou být uvedeny modifikátory přístupu

- V deklaraci parametru může být uvedena defaultní hodnota(např.int hodnota=4)

- V deklaraci každého parametru musí být vždy uveden typ

- Pokud má metoda více parametrů stejného typu, lze uvést typ parametru pouze jednou, např, public void metoda(int x, y) { ... }

Pravdivý výrok o statických metodách:

- Statickou metodu deklarovanou ve třídě A lze volat i z metod instancí A

(tohle není otázka, jenom správná odpověď)

Vyberte pravdivý výrok o statické konstantě (static final):

- přiřazení hodnoty do statické konstanty musí být součástí deklarace této konstanty

- statická konstanta může být pouze primitivního datového typu, např.public static final double PI = 3.14;

- neměnná statická konstanta musí mít modifikátor přístupu public

- statické konstanty mohou být i referenčního typu

- Nadeklarujeme následující statickou konstantu public static final int

DEN_NEDELE = 0; Při tisku pomocí System.out.println

("" + DEN_NEDELE) se vytiskne řetězec "DEN_NEDELE" (vytiskne se číslo)

Vyberte pravdivý výrok o statických proměnných:

- statická proměnná je v paměti pouze jednou bez ohledu na počet vytvořených instancí

- statická proměnná může být pouze primitivního datového typu, popř. pole primitivního datového typu

- statická lokální proměnná může být deklarována uvnitř statické metody, nemůže být v metodě instance

- uvnitř metody instance může být deklarována statická lokální proměnná

- privátní statická proměnná deklarovaná ve třídě A není dostupná z metod instancí třídy A. Je dostupná pouze ze statických metod třídy A

Vyberte nepravdivý výrok o statických proměnných:

- K obsahu privátní statické proměnné lze přistupovat z konstruktoru stejné třídy

- Statická proměnná musí mít v deklaraci uveden modifikátor static

- Statická proměnná třídy může mít uvedeny všechny modifikátory přístupu (public, protected, private, „přátelský přístup")

- Statickou proměnnou lze deklarovat uvnitř statické metody (statická lokální proměnná)

Vyhrazené slovo „null" se v Javě používá pro:

- označení obsahu proměnné referenčního typu, který neobsahuje odkaz na nějakou Instanci

(tohle není otázka, jen správná odpověď)

Vyhrazené Slovo „this" má v Javě následující využití:

- Při volání metod stejné instance lze this použít na místě odkazu na instanci

- Slovo this může být uvedeno jako návratová hodnota metody

- Slovo this označuje situaci, kdy proměnná referenčního typu neobsahuje odkaz na nějakou existující instanci

- This je operátor přetypování referenčních typů

Vyberte situaci, kdy "nelze" použít klíčové slovo public

- v deklaraci (hlavičce) metody

- v deklaraci (v hlavičce) třídy

- v deklaraci parametru metody

- v deklaraci datového atributu

Vyberte správný způsob zobrazení použití výčtového typu DenVTydnu ve třídě Rozvrh v diagramu UML

- „Rozvrh"a <<enum>>DenVTydnu - pozor na šipku

(tohle není otázka, jen odpověď)

Která tvrzení o mapách (třídách implementujících rozhraní Map<K,V>) jsou pravdivá a která ne?

- Dvě hodnoty v mapě mohou být shodné podle metody (equals)

- Mapa typu HashMap obsahuje dvojice setříděné podle hodnoty klíče

- Dva klíče v mapě mohou být shodné (podle metody equals)

- Z mapy je pomocí metody keySet() možné získat množinu klíčů

- Třída, jejíž instance jsou klíče v mapě typu HashMap, by měla mít implementovány metody equals() a hashCode().

- Klíčem v mapě mohou být I hodnoty primitivních datových typů

- Prvkem mapy je dvojice tvořená klíčem a k němu připojenou hodnotou.

Která tvrzení o množinách(třídách implementujících rozhraní Set<E>)jsou pravdivá a která ne?

- Počet prvků v množině lze zjistit pomocí metody size()
- Prvek vkládaný do množiny pomocí metody add() je vložen vždy na konec
- Množinu je možné procházet pomocí cyklu for - each, ale pomocí klasického for ne.
- Třída, jejíž instance jsou vkládány by měla mít hashcode + equals

Která tvrzení o polích v Javě jsou pravdivá a která ne?

- Pole nemůže obsahovat prvky primitivních datových typů.
- Pole o N prvcích má indexy od 0 do N-1.
- Rozsah pole (jeho prvků) se určuje už při deklaraci
- Velikost jednorozměrného pole lze zjistit pomocí proměnné length
- Všechny prvky pole jsou stejného typu nebo jeho podtypů.
- Dvourozměrné pole musí mít stejný počet "sloupců" ve všech "řádcích"

Máme dvě třídy Hamburger a BigMac, třída BigMac je potomkem třídy Hamburger.

Jsou definovány dvě proměnné takto:

Hamburger hamburger;

BigMac bigMac;

Která z následujících přiřazení jsou legální (tj. přeloží se bez chyby) a která ne?

hamburger = new Hamburger();

- bigMac = hamburger;
- hamburger = new BigMac();
- bigMac = new Hamburger();
- bigMac = new BigMac();
- hamburger = bigMac;

Máme dvě třídy Zvire a DomaciZvire, třída DomaciZvire je potomkem tridy zvire

Jsou definovány dvě proměnné takto: Zvire kocka; DomaciZvire micka. Která se OK přeloží?

- kocka = new DomaciZvire()
- micka = new Zvire()
- kocka = micka

- micka = kocka

- kocka = new Zvire()

- micka = new DomaciZvire()

Některé metody (nemluvíme o konstruktoru) nemusí vracet žádnou návratovou hodnotu

- programátor nesmí do metody napsat příkaz return

- Metoda má v hlavičce uvedeno slovo void jako typ návratové hodnoty.

- programátor musí do vnitřku metody napsat příkaz return

- metoda musí být veřejně dostupná (public)

Označte pravdivá a nepravdivá tvrzení o abstraktních třídách.

- ...uveden modifikátor abstrakt...

- Abstraktní metody mohou být deklarovány pouze v abstraktní třídě nebo rozhraní.

- Abstraktní třída nesmí mít potomky.

- Abstraktní třída nemá konstruktor.

- Instanci abstraktní třídy nelze vytvořit pomocí konstruktoru této třídy

- Abstraktní třída musí obsahovat alespoň jednu abstraktní metodu.

Označte pravdivá a nepravdivá tvrzení o rozhraní v Javě:

- Všechny metody deklarované v rozhraní jsou veřejné (public).

- Rozhraní má vždy pouze implicitní konstruktor

- Rozhraní může obsahovat pouze statické konstanty, deklarace metod a případně vnořené (static inner, nested) třídy

- V deklaraci rozhraní musí být uvedeno klíčové slovo interface.

- Dynamická sémantika ... // nevím co je tím myšleno, ale princip se tam vyskytuje

Označte pravdivé a nepravdivé výroky.

- Aritmetické operátory *, / a % mají stejnou prioritu.

- Aritmetické výrazy bez závorek se vyhodnocují zleva doprava.

- Pokud se v Javě deklaruje proměnná, musí být vždy uveden její typ

- V Javě jsou identifikátory number a Number identické

Označte pravdivost jednotlivých výroků o syntaxi a sematice programovacích jazyků:

- Statická sémantika je ta část sémantiky,kterou lze kontrolovat při překladu-např. pravidla pro typovou kontrolu.

- Pro popis syntaxe síťových protokolů (např. protokolu HTTP) se používá upravená verze Backus-Naurovy formy. //verze ABNF

- V Backus-Naurově formě (BNF) lze popsat i sémantiku jazyka.

- Dynamická sémantika-obvykle popisována běžným jazykem

- Backus-Naurova forma (BNF) se používá pro popis syntaxe jazyka.

- Komentáře se ze zdrojového kódu odstraňují při lexikální analýze.

- Pro popis sémantiky jazyka se používají bezkontextové gramatiky.

Označte pravdivost jednotlivých výroků o polích v programovacích jazycích:

- Java podporuje pouze pravoúhlá dvourozměrná pole (java-zubatá pole)

- V zubatém vícerozměrném poli může mít každý řádek rozdílný počet prvků

- V Javě se alokuje paměť pro pole dynamicky při deklaraci pole (pevná velikost pole je určená při inicializaci)

- Pole je homogenní datová struktura, ve které se pozice prvku udává pomocí indexu - ArrayList v Javě splňuje tuto definici

- Příkladem jazyka se statickou alokací paměti pro pole (tj. Při překladu) je Fortran

Označte pravdivost jednotlivých výroků o procedurálních programovacích jazycích:

- Procedurální programovací jazyky neobsahují syntaktická omezení pro použití procedur (metod, funkcí).

- Všechny procedurální jazyky umožňují vytvářet vlastní abstraktní datové typy.

- Procedurální jazyky se používají pro opakující se operace cykly.

- Všechny procedurální programovací jazyky obsahují příkaz goto.

Označte pravdivost tvrzení o komentářích

- Program javadoc zpracovává pouze víceřádkové komentáře, které začínají /** (a nejsou uvnitř metody).

- Je možné uvést komentář i uvnitř výrazu, např. int i = 3 + / komentar / 25;

- Na začátku každého meziřádku ve víceřádkovém komentáři musí být uvedena *

- Víceřádkový komentář začíná / a končí /

- Jednořádkové komentáře lze uvést pouze uvnitř metod (a konstruktorů).

Označte pravdivost tvrzení týkající se jazyka XML

- Pokud chceme převést XML dokument do jiného formátu (např. PDF či HTML), můžeme použít popis transformace popsaného v XSD (XML Schema Definition Language).

- Parsovaní označuje proces kontroly datové struktury XML dokumentu a následné jeho rozložení do malých jednotlivých částí, se kterými může pracovat aplikace.

- DTD (Document Type Definition) se používá pro popis datové struktury konkrétního XML dokumentu, tj. jaké jsou přípustné značky, elementy a atributy.

- Data v XML dokumentu vytvářejí stromovou strukturu.

Označte pravdivost následujících výroků o typech proměnných v programovacích jazycích:

- Pojem typová inference označuje přístup, kdy se typ parametrů metody a typ návratové hodnoty odvozuje z obsahu metody.

- Typová nezávislost označuje situaci, kdy do proměnné můžeme přiřadit hodnotu libovolného typu.

- U "silně typových jazyků" jsou všechny chyby v typech zjištěny při překladu či za běhu programu.

- Java provádí všechnu typovou kontrolu při překladu a žádnou za běhu.

Označte pravdivost tvrzení o životnosti proměnných v programovacích jazycích:

- Životnost proměnné označuje dobu, kdy proměnná má přiřazenu paměť.

- Všechny proměnné s modifikátorem static v Javě mají přidělenu paměť od spuštění programu do konce programu.

- Parametrům metod a lokálním proměnným metod se obvykle přiděluje paměť v zásobníku (stack).

- Datové atributy typu int v Javě mají shodné hranice rozsahu platnosti a životnosti.

Označte pravdivost výroků o rozsahu platnosti proměnných v jazycích:

- Rozsah platnosti proměnné vymezuje hranice, ve kterých se lze na proměnnou odkázat.

- Přetypování instancí v Javě je ukázkou dynamického rozsahu platnosti proměnné.

- Konstanta PI ze třídy Math v Javě má neomezený rozsah platnosti (lze používat všude).

- Java používá statický rozsah platnosti kontrola se provádí při překladu.

Třída MojeTrida implementuje rozhraní Rozh1 a Rozh2.Která z následujících přiřazení jsou správná

- Rozhr1 rozhrani1 = new MojeTrida(); Rozhr2 rozhrani2 = rozhrani1

- Moje Trida Instance1 = new MojeTrida();

- Rozhr1 rozhrani1 = new Rozhr1()

- Rozhr1 rozhrani2 = new Rozhr2()

- Rozhr1 rozhrani1 = new MojeTrida();

- Rozhr2 rozhrani2 = new MojeTrida();

- Moje Trida Instance1= newMojeTrida(); Rozhr1 rozhrani1=instance1;MojeTrida instance2=(MojeTrida)rozhrani1;

- MojeTrida Instance 1 =new MojeTrida(), Rozhr1 rozhrani1=instance1

- Rozhr1 rozhrani1 = new MojeTrida(); Rozhr2 rozhrani2 = rozhrani1

- Rozhr2 rozhran2 = new Rozhr2()

- Rozhr2 rozhrani2 = new MojeRozhr2()

- Rozhr1 rozhrani1 = new MojeRozhr1()

Vyberte pravdivá tvrzení o rekurzi při programování:

- Při rekurzi se obvykle z metody A volá metoda A, tj. metoda volá sama sebe (tzv. rekurzivní metoda).

- Rekurze znamená, že pro řešení problému využijeme menších instancí stejného problému.

- Rekurzivní metody nemohou mít vedlejší efekty, tj. nemohou měnit i jinou proměnnou, než je návratová hodnota.

- Nepřímá rekurze je situace, kdy vzájemné volání metod vytvoří „kruh". Např. z metody A je volána metoda B, z metody B voláme metodu C, která volá metodu A.

Co se nepřeloží?

double cislo = 4.53;

- if (cislo > 4.53)

- if (cislo <> 5.43)

- if (cislo == 4.5)

- if (cislo > 43L)

- if (cislo != 5.43)

Dědičnosti v Javě...

- V javě je možná pouze jednonásobná dědičnost (tj. Třída může mít pouze jednoho přímého předka)

- Potomek dědí pouze to co není private

- Dědí se i konstruktory (tj. Pokud má předek konstruktor s jedním parametrem typu String, automaticky ho má i potomek)

- Zděděné metody lze v potomkovi překrýt

Seřadte jednotlivé fáze překladu programu u klasického překladače:

- Sématická analýza 2

- Lexikální anal. 1

- Linkovnání programu s knihovnami 5

- Syntaktická analýza 3

- Generování 4

Záhlaví třídy (samos. ne Ve vnitřním bodu) může obs.

- Public

- extends

- impements

Přiřaď zprávnou definici ke znaku konverze či příznaku formátování

znaky konverze:

h ; d ; 0 ; g ; , ; x ; t ; - ; + ; b ; f ; s

Definice:

- text, výsledek toString(), popř. null

- boolean

- zarovnání vlevo

- celé číslo hex

- desetinné číslo, u velkých čísel vědecký formát

- celé číslo v dekadickém tvaru

- desetinné číslo

- u čísel i znaménko +

- formátování datumu a času

- vloží se oddělovač řádů dle národního prostředí

- vypíše se úvodní nula

- celé číslo v hex

s - text, výsledek toString(), popř. null
b - boolean
- - zarovnání vlevo
x - celé číslo hex
g - desetinné číslo, u velkých čísel vědecký formát
d - celé číslo v dekadickém tvaru
f - desetinné číslo
+ - u čísel i znaménko +
t - formátování datumu a času
, - vloží se oddělovač řádů dle národního prostředí
0 - vypíše se úvodní nula
h - celé číslo v hex

Vyberte správné tvrzení platící o třídě

- Obecný popis obsahujicí data, které popisují stav objektu

- Vždy implementuje rozhraní pro danou třídu charakteristické

- Obsahuje metody, které popisují činnosti, které je možné s objektem provádět

- Je definovaná návratovou hodnotou a identifikátorem přístupu