

```
public class R {  
    public double x;  
    public double y;  
  
    public R() {  
        this(3.5);  
    }  
  
    public R(int x) {  
        this(x,x);  
    }  
  
    public R(int x, int y) {  
        this(x,y);  
    }  
  
    public R(double x, double y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

Što će biti rezultat prevodenja napisanog izvornog koda u byte-kod?

a

Prevodilac će prijaviti pogrešku da ne mogu postojati dva konstruktora koja oba primaju dva argumenta

b

Kod će se uredno prevesti i nastat će class-datoteka

c

Prevodilac će prijaviti pogrešku kod konstruktora R(int x, int y)

d

Prevodilac će prijaviti pogrešku kod konstruktora R(int x)

a

izvedena klasa iz tipa koji je definiran u toj metodi

b

tip definiran u toj metodi

c

bilo koji tip

d

sve od navedenog

```
interface I1 { }
class C1 implements I1 { }
public class Main {
    public static void main(String[] args) {
        I1 a = new C1(); // (1)
        C1 b = new C1(); // (2)
        b = a; // (3)
        a = b; // (4)
        a = new I1(); // (5)
    }
}
```

a

C1 b = new C1(); // (2)

b

b = a; // (3)

c

a = b; // (4)

d

I1 a = new C1(); // (1)

e

a = new I1(); // (5)

Deklarirani su sljedeći tipovi:

```
interface I1 { }
interface I2 { }
class C1 { }
class C2 { }
```

Označite sve deklaracije koje **nisu** dozvoljene i javljaju grešku tijekom prevođenja?

a

```
interface I4 extends I1, I2 { }
```

b

```
class C4 extends I1, I2 { }
```

c

```
class C5 extends C1 implements I1, I2 { }
```

d

```
class C3 implements I2 { }
```

e

```
interface I3 extends I1 { }
```

Nadklasa sadrži metode zajedničke svim izvedenim klasama u hijerarhiji, ostavljajući mogućnost da pojedina izvedena klasa nadjača metodu svojom specifičnom implementacijom. Takve metode u izvedenoj klasi nazivamo:

a public metode

b dinamičke metode

c virtualne metode

d statičke metode

Za klase koje su označene s final vrijedi:

a

ne mogu se naslijediti

b

ne mogu se napraviti objekti iz njih

c

mogu se napraviti objekti iz njih

d

klasa koja ih nasljeđuje ne može promijeniti niti jedan konstruktor

e

klasa koja ih nasljeđuje mora nadjačati neke metode

U programskom jeziku Java, koji se modifikator vidljivosti može postaviti za klasu koja se nalazi u istoimenoj datoteci?

a public

b private

c java.class

d protected

e package

```
--  
for(int i = 0; i < r.length; i++) {  
    new r[i] = R(i,i);  
}  
r[1].x++;  
System.out.println(r[0]);  
System.out.println(r[1]);
```

Što će biti rezultat pokretanja programa?

a

program se zbog pogrešaka u metodi main neće prevesti

b

program se zbog pogrešaka u razredu R neće prevesti

c

ispis 1,0 pa ispis 1,0

d

ispis 0,0 pa ispis 2,1

Ako imamo definirane sljedeće stringove

```
String s1 = new String("jedan");
String s2 = "jedan";
```

što će se ispisati naredbom `System.out.print(s1==s2);`

a

true

b

false

c

`s1==s2`

d

jedan

e

Program će izbaciti pogrešku

Što od navedenog vrijedi za poziv apstraktne metode?

- a** primjerak potklase se može castati na tip ove klase
- b** mora imati barem jednu apstraktну metodu
- c** s konstruktorom se može kreirati primjerak (instanca) ove klase
- d** ništa od navedenog

Što sve od navedenog je vrsta polimorfizma?

a

implicitni poliformizam

b

naslijedni polimorfizam

c

hijeracijski polimorfizam

d

ad-hoc polimorfizam

U paketu hr.fer.oopj definiran je razred R kako slijedi:

```
public class R {  
    public int x;  
    public int y;  
  
    public R(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    public String toString() { return x+","+y; }  
}
```

Potom je u metodi main nekog drugog razreda napisano:

```
R[] r = new R[3];  
for(int i = 0; i < r.length; i++) {  
    new r[i] = R(i,i);  
}  
r[1].x++;  
System.out.println(r[0]);  
System.out.println(r[1]);
```

Što će biti rezultat pokretanja programa?



program se zbog pogrešaka u metodi main neće prevesti

program sa zvaničnog recnika u razred R neće provesti

Prilikom nadjačavanja neke metode, povratna vrijednost može biti samo ona definirana u toj metodi.

a točno

b netočno

U paketu hr.fer.oopj definiran je razred R kako slijedi:

```
public class R {  
    public double x;  
    public double y;  
  
    public R() {  
    }  
  
    public R(double x, double y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

U metodi main smještenoj u razredu X u istom paketu želimo deklarirati lokalnu varijablu r i pridružiti joj jedan novostvoren objekt razreda R. Koji će od ponuđenih izraza to učiniti?

a R r = malloc(sizeof(R));

b R r = new R;

c R r = new R();

d R r = new R(3.14);

Što treba raditi metoda equals iz klase Object koju nadjačavamo:

- a uspoređuje da su reference spremljene u istu varijablu
- b uspoređuje da reference pokazuju na objekte koji imaju isti sadržaj
- c uspoređuje da objekti imaju isti reprezentaciju sadržaja
- d uspoređuje da reference pokazuju na isti objekt
- e uspoređuje da su objekti spremljeni u istu varijablu

Ako u nekoj baznoj klasi postoji metoda calculatePrice vidljivosti protected, koje SVE vidljivosti može biti metoda calculatePrice iz izvedene klase kojom se nadjačava?

a Ništa od navedenog

b private

c public

d protected

U kojem slučaju kada se događa boxing?

```
int num = 10; // 1  
Integer base = num; // 2  
for (int i = 0; i < 3; i++, base++) { // 3  
    base = base + i; // 4
```

J

a 2

b 1

c 4

d 3

Ako u programskom obrisniku našlik dvojog skoci metoda M1 baca iznimku tipa *NullPointerException* koje od naredbi će se izvršiti

```
try {
    M1();
    System.out.println("T1");
} catch (IllegalArgumentException exc) {
    System.out.println("CATCH");
}
finally {
    System.out.println("F");
}
System.out.println("T2");
```

a T2

b F

c T1

Koja je izjava točna, a odnosi se na niz naredbi koje su upisane u main metodu?

```
short s1 = 100;          // 1
Short s2 = s1;           // 2
Object o = s2;           // 3
Number n = o;            // 4
System.out.println(n);   // 5
```

a

pogreška prevoditelja u liniji 1

b

pogreška prevoditelja u liniji 3

c

kod će se uspješno prevesti

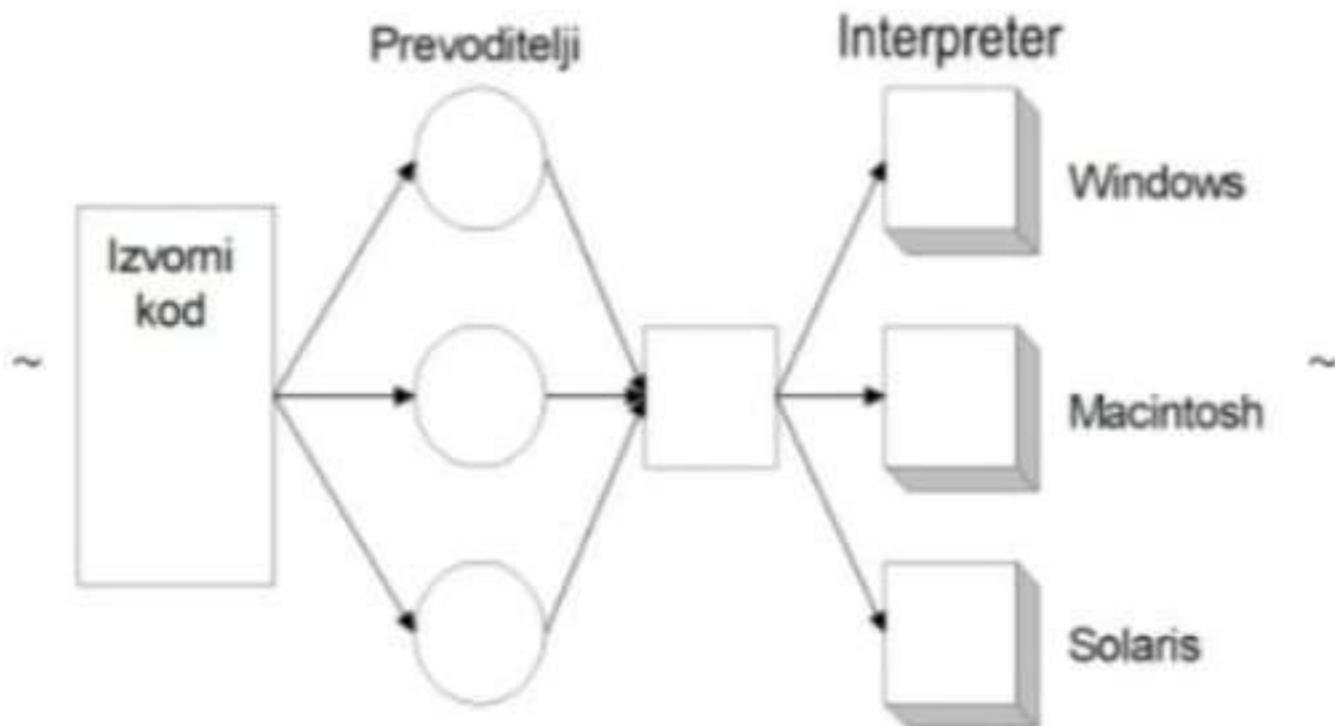
d

pogreška prevoditelja u liniji 2

e

pogreška prevoditelja u liniji 4

Prilikom prevodenja Javiniog izvornog koda stvara se



a

java-code

b

interpreter-code

c

byte-code

```
public class Point<T> {  
    ...  
    public <V> void printWith(V another) {  
        // što je T, a što je V u ovom trenutku?  
        System.out.format("first: %s second %s %n", this.toString(), another.toString());  
    }  
}
```

Označite točne odgovore vezane za pitanje `što je T, a što je V u ovom trenutku?` ako metodu `printWith` pozovemo iz sljedećeg programskog odsječka:

```
Point<Integer> x = new Point<>(2, 3);  
Point<String> y = new Point<>("A", "B");  
x.printWith(y); // što je T, a što V prilikom ovog poziva metode?
```

a T je `Point<Integer>`

b V je `Point<String>`

c T je isto što i V

d V je `String`

e T je `Integer`

Neka je razred R resurs u smislu kako to definira naredba try-with-resources. Razmatramo jednu konkretnu naredbu try-with-resources koja koristi takve resurse. Što je od sljedećega točno?

- a** Reference na korištene resurse dostupne su nam u blokovima catch i finally
- b** Kada koristimo resurse, ne smijemo pisati blok finally
- c** Resursi se stvaraju u oblim zagradama odmah nakon ključne riječi try
- d** Resursi se stvaraju u vitičastim zagradama odmah nakon ključne riječi try
- e** Resursi se stvaraju u oblim zagradama nakon što se u vitičastim zagradama napisanim nakon ključne riječi try definira programski kod

Klase Throwable omogućava pristup podatcima kao što su:

- a ništa od navedenog
- b pristup do "omotane" iznimke, ako takva postoji
- c točna lokacija iznimke u kodu (koja datoteka, koji redak) za svaku metodu
- d poruka pogreške
- e cjelokupno stanje na stogu u trenutku kada je nastala iznimna situacija

Ako je klasa Gen parametrizirana po tipu T (tj. `class Gen<T>`) te je zadana metoda `T m(T t)`, koje od sljedećih naredbi su moguće u metodi m?

a

```
return (T[]) new Object[10];
```

b

```
return new T();
```

c

```
return t;
```

d

```
System.out.println(t.toString());
```

e

```
t = (T[]) new Object[10];
```

Neka je razred R resurs u smislu kako to definira naredba try-with-resources. Razmotrite sljedeći kod napisan u metodi main.

```
try (R r = new R()) { }
finally { }
```

Što je od sljedećega točno?

- a** kod se neće prevesti jer ne smije imati blok finally
- b** kod se neće prevesti jer nismo zatvorili resurs
- c** kod se neće prevesti jer nedostaje blok catch
- d** try-konstrukt je ispravan i moći će se prevesti

Označite točne tvrdnje

- a Klasa *Exception* i njeno podstablo modelira opise iznimnih situacija od kojih se očekuje da je moguć oporavak.
- b Klasa *Error* i njeno podstablo modelira opise iznimnih situacija od kojih se **ne** očekuje da je moguć oporavak.
- c Klasa *Error* i njeno podstablo modelira opise iznimnih situacija od kojih se očekuje da je moguć oporavak.
- d Klasa *Exception* i njeno podstablo modelira opise iznimnih situacija od kojih se **ne** očekuje da je moguć oporavak.

Klasa IntegerPoint je definirana kako je navedeno. Pomoću kojih naredbi možemo kreirati objekt tipa IntegerPoint?

```
public class IntegerPoint {  
    private Integer x, y;  
    public IntegerPoint(Integer x, Integer y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

a `IntegerPoint ip = new IntegerPoint(Integer.valueOf(2), Integer.valueOf(3));`

b `IntegerPoint ip = new IntegerPoint(2, 3);`

c `IntegerPoint ip = new IntegerPoint("2", "3");`

d `IntegerPoint ip = new IntegerPoint(2L, 3L);`

e `IntegerPoint ip = new IntegerPoint(new Integer(2).intValue(), new Integer(3).intValue());`

Koji će biti rezultat razvoja (prevođenje i pokretanje) jednostavne Java aplikacije? Aplikaciju prevodimo i pokrećemo naredbom

```
javac Proba.java  
java Proba "Ana Marija"
```

pri čemu je sadržaj datoteke Proba.java sljedeći:

```
public class Proba {  
    static int god = 2001;  
    public static void main(String[] argv) {  
        System.out.format("%s je rođena %d. godine.", argv[0], god);  
    }  
}
```

a

Uspješno prevođenje i pokretanje s ispisom: Ana Marija je rođena 2001. godine.

b

Uspješno prevođenje i pokretanje s ispisom: Marija je rođena 2001. godine.

c

Uspješno prevođenje i pokretanje s ispisom: Ana je rođena 2001. godine.

d

Uspješno prevođenje, ali pogreška tijekom izvođenja

e

Pogreška tijekom prevođenja

Neka je razred R resurs u smislu kako to definira naredba try-with-resources. Razmatramo jednu konkretnu naredbu try-with-resources koja koristi takve resurse. Što je od sljedećega točno?

- a Resursi se stvaraju u vitičastim zagradama odmah nakon ključne riječi try
- b Kada koristimo resurse, ne smijemo pisati blok finally
- c Reference na korištene resurse dostupne su nam u blokovima catch i finally
- d Ako naredba treba stvoriti više resursa i jedan od njih izazove iznimku, prethodno stvoreni resursi bit će zatvoreni
- e Resursi se stvaraju u oblim zagradama nakon što se u vitičastim zagradama napisanim nakon ključne riječi try definira programski kod

Ispravno pokretanje programa *Example* koji se nalazi u paketu *hr.fer.oop.primjeri*, prevedenom u direktoriju *bin*, u konzoli se vrši naredbom:

a jar hr.fer.oop.primjeri.Example

b java -cp bin hr.fer.oop.primjeri.Example.class

c jar -cp bin hr.fer.oop.primjeri.Example

d java -cp bin hr.fer.oop.primjeri.Example

e java -cp bin hr/fer/oop/primjeri/Example.java

Koji od ponuđenih naziva predstavljaju klase omotačkih objekata?

a

Char

b

Long

c

Numeric

d

Short

e

Byte

U programskom jeziku Java iznimne situacije smo na predavanju podijelili na velike porodice kojima pripadaju:

a

provjeravane iznimke

b

grube pogreške

c

neprovjeravane iznimke

d

greške povezivanja

e

sintaksne iznimke

Ako je klasa Gen parametrizirana po tipu T, (tj. početak definicije klase glasi `class Gen<T>`) što od navedenog je moguće definirati unutar klase Gen?

a

```
static <V> void mv(V v) { }
```

b

```
static void m5(T t) { }
```

c

```
static T m4(T t) { return t; }
```

d

```
Gen<Gen<T>> g;
```

e

```
static <V> V m(V v) { return v; }
```

Neka je razred R resurs u smislu kako to definira naredba try-with-resources. Razmatramo jednu konkretnu naredbu try-with-resources koja koristi takve resurse. Što je od sljedećega točno?

- a Kada koristimo resurse, ne smijemo pisati blok finally
- b Ako naredba treba stvoriti više resursa i jedan od njih izazove iznimku, resursi deklarirani nakon tog resursa neće biti stvorenii
- c Reference na korištene resurse dostupne su nam u blokovima catch i finally
- d Resursi se stvaraju u vitičastim zagradama odmah nakon ključne riječi try
- e Resursi se stvaraju u oblim zagradama nakon što se u vitičastim zagradama napisanim nakon ključne riječi try definira programski kod

Što ispisuje sljedeći program?

```
public class Ispis {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = new String(s1);  
        s2.toUpperCase();  
        System.out.println((s1 == s2) + " " + s1.equals(s2));  
    }  
}
```

a

false false

b

true false

c

false true

d

true true

Koja je izjava točna a odnosi se na niz naredbi koje su upisane u main metodu?

```
Double d1 = Double.parseDouble("3.14");      // 1
double d2 = d1.doubleValue();                  // 2
double d3 = d1;                             // 3
Double d4 = new Double(0.);                  // 4
d4.setValue(1.11);                         // 5
```

a

prevoditelj javlja pogrešku u liniji 5

b

prevoditelj će javiti pogrešku u liniji 1

c

prevoditelj će javiti pogrešku u liniji 3

d

prevoditelj će javiti pogrešku u liniji 4

e

prevoditelj će javiti pogrešku u liniji 2

U programskom jeziku Java, koje od ponuđenih modifikatora vidljivosti možemo postaviti metodama klase A, kako bi te metode bile vidljive klasi B koja je u istom paketu kao i klasa A?

a može biti bez modifikatora

b private

c protected

d public

e import

```
try{  
    M1();  
    System.out.println("T1");  
}catch (NullPointerException exc){  
    System.out.println("CATCH");  
}  
finally {  
    System.out.println("F");  
}  
System.out.println("T2");
```

a

F

b

CATCH

c

T1

d

T2

Neka je razred R resurs u smislu kako to definira naredba try-with-resources. Razmatramo jednu konkretnu naredbu try-with-resources koja koristi takve resurse. Sto je od sljedećega točno?

- a** Ne trebamo pisati programski kod u bloku finally koji brine o zatvaranju resursa
- b** Reference na korištene resurse dostupne su nam u blokovima catch i finally
- c** Resursi se stvaraju u oblik zagradama nakon što se u vitičastim zagradama napisanim nakon ključne riječi try definira programski kod
- d** Resursi se stvaraju u vitičastim zagradama odmah nakon ključne riječi try
- e** Kada koristimo resurse, ne smijemo pisati blok finally

Neka je vršni direktorij projekta `C:\projekt1`. Prema naputku za organizaciju strukture direktorija koji je dan na predavanju, gdje bi trebao biti smješten izvorni kod razreda definiranih u paketu `hr.fer.oop.primjer?`

a

`C:\projekt1\primjeri\oop\fer\hr\src`

b

`C:\projekt1\src\hr\fer\oop\primjeri`

c

`C:\projekt1\hr\fer\oop\primjeri`

d

`C:\projekt1\bin\hr\fer\oop\primjeri`

e

`C:\projekt1\primjeri\oop\fer\hr\bin`

Koji od navedenih tipova iznimki spadaju u neprovjeravane iznimke?

a

IOException

b

IndexOutOfBoundsException

c

NumberFormatException

d

SQLException

e

NullPointerException

Neka je razred R resurs u smislu kako to definira naredba try-with-resources (drugim riječima, implementira sučelje java.lang.AutoCloseable pa njegove objekte na kraju uporabe treba zatvoriti pozivom metode close()). Razmotrite sljedeći kod napisan u metodi main.

```
try {  
    R r = new R();  
    r.close();  
}
```

Što je od sljedećega točno?

a try-konstrukt nije ispravan jer mora imati barem jedan blok catch ili finally

b kod se neće prevesti jer u bloku try imamo više od jedne naredbe

c try-konstrukt je ispravan i moći će se prevesti

d kod se neće prevesti jer ne smijemo sami zatvarati resurs

Ako je klasa Gen parametrizirana po tipu T, (tj. početak definicije klase glasi `class Gen<T>`) te postoji parametrizirana klasa *MyList* što od navedenog je moguće definirati unutar klase *Gen*?

a

```
static myList<T> list;
```

b

```
<V> void m4(T t, Gen<Gen<V>> v) { }
```

c

```
myList<myList<T>> listofList;
```

d

```
static <V> void m(myList<V> list) { }
```

e

```
T obj1;
```

Stablo iznimki počinje od klase:

a

RuntimeException

b

Exception

c

AssertionError

d

Error

e

Throwable

f

LinkageError

Koja se metoda prva poziva prilikom pokretanja programa?

- a** Main metoda
- b** Ona koja je prva po abecednom redu
- c** Ona koja je prva napisana
- d** Naziv metode koju treba pokrenuti navodi se iz naredbenog retka prilikom pokretanja programa
- e** Nasumično, bilo koja

Ako je klasa Gen parametrisirana po tipu T (tj. `class Gen<T>`), koja od sljedećih naredbi bi bila ispravna u nekoj od metoda klase Gen?

a

```
T[] arr = (T[]) new Object[50];
```

b

```
T[] arr = new T[];
```

c

```
T[] arr = new T[50];
```

d

```
T[] arr = (Object[]) new T[50];
```

e

```
T[] arr = new Object<T>[50];
```

Što sve od navedenog vrijedi za neprovjeravane iznimke?

a

Mora ih se eksplicitno obrađivati ili naglasiti da se prosleđuju dalje

b

Modeliraju situacije za koje želimo da se eksplicitno obrađuju

c

Modeliraju situacije koje se mogu javiti na svakom koraku izvođenja programa

d

Nije nužno ugraditi upravljanje takvim iznimkama

e

Takve iznimke nisu fatalne za aplikaciju (oporavak je moguć)

U kojim linijama koda se događa *unboxing*?

```
Integer i = new Integer(32);           // 1
if(i > 30) {                         // 2
    System.out.println("i > 30");      // 3
    i++;                                // 4
}
```

a

4

b

1

c

3

d

2

Označite točne odgovore vezane za pitanje što je T, a što je V u ovom trenutku? ako metodu *printWith* pozovemo iz sljedećeg programskog odsječka:

```
Point<Integer> x = new Point<>(2, 3);
Point<String> y = new Point<>("A", "B");
x.printWith(y); // što je T, a što V prilikom ovog poziva metode?
```

a T je Integer

b V je String

c V je Point<String>

d T je Point<Integer>

e T je isto što i V

```
try {
    try(Spremnik s1 = new Spremnik("pero"); Spremnik s2 = new Spremnik("ana")) {
        System.out.print("N1;");
        s1.write(5);
        System.out.print("N2;");
        s2.write(0);
        System.out.print("N3;");
    }
    catch(ArithmeticException ex) {
        System.out.print("AE;");
    }
    finally {
        System.out.print("FI1;");
    }
}
catch(RuntimeException ex) {
    System.out.print("RE;");
}
finally {
    System.out.print("FI2;");
}
```

Razmotrite sljedeći kod koji definira klasu Spremnik. Iznimke ArithmeticException i IllegalArgumentException izvedene su iz iznimke RuntimeException.

```
public class Spremnik {  
    private String ime;  
    public Spremnik(String ime) {  
        if(ime.length()==0)  
            throw new IllegalArgumentException();  
        this.ime = ime;  
        System.out.print("Stvoren " + ime + ";");  
    }  
    public void close() {  
        System.out.print("Zatvoren " + ime + ";");  
    }  
    public void write(int i) {  
        if(i<0) throw new RuntimeException();  
        if(i==0) throw new ArithmeticException();  
    }  
}
```

Ako je u metodi main nekog razreda napisan sljedeći kod, što će biti ispisano kao rezultat njegova izvođenja?

```
try {  
    try(Spremnik s1 = new Spremnik("pero"); Spremnik s2 = new Spremnik("ana")) {  
        System.out.print("N1");  
        s1.write(5);
```

a

Stvoren pero;Stvoren ana;N1;N2;N3;FI1;FI2;

b

Ništa. Program se neće prevesti jer nakon oble zagrade iza ključne riječi try ne možemo stvarati primjerak klase Spremnik.

c

Stvoren pero;Stvoren ana;N1;N2;AE;FI1;FI2;

d

Stvoren pero;Stvoren ana;N1;N2;N3;Zatvoren ana;Zatvoren pero;FI1;FI2;

e

Stvoren pero;Stvoren ana;N1;N2;Zatvoren ana;Zatvoren pero;FI1;RE;FI2;

f

Stvoren pero;Stvoren ana;N1;N2;Zatvoren ana;Zatvoren pero;AE;FI1;FI2;

g

Stvoren pero;Stvoren ana;N1;N2;Zatvoren ana;Zatvoren pero;AE;FI1;RE;FI2;

Automatsko upravljanje resursima može se koristiti s klasama koje implementiraju neko od sljedećih sučelja:

a Closeable

b Throwable

c AutoCloseable

d Finally

e Resource

U programskom jeziku Java, modifikator *public* (koji se nalazi ispred metode) omogućuje sljedeće pristupe:

a naslijeđene klase

b javni pristup

c klase unutar istog paketa

d ništa od navedenog

e unutar klase

Koja je izjava točna, a odnosi se na niz naredbi koje su upisane u main metodu?

```
double d1 = 10.5;           // 1
Double d2 = d1;             // 2
Object o1 = d2;             // 3
Integer i1 = (Integer) o1; // 4
System.out.println(i1);     // 5
```

- a** pogreška prevoditelja u liniji 2
- b** kod će se uspješno prevesti ali će se dogoditi *ClassCastException* tijekom izvođenja
- c** pogreška prevoditelja u liniji 4
- d** kod će se uspješno prevesti i nakon pokretanja ispisat će se 10
- e** pogreška prevoditelja u liniji 1

Ako je klasa Gen parametrizirana po tipu T, (tj. početak definicije klase glasi `class Gen<T>`) što od navedenog je moguće definirati unutar klase *Gen*?

a

```
T n(T t){ return t;}
```

b

```
static T obj2;
```

c

```
T obj1;
```

d

```
void o(Integer i) { }
```

e

```
void m(T t) { }
```

U kojim od sljedećih slučajeva instanciranje parametrizirane klase Point je napravljeno ispravno:

a

```
Point<> iPoint = new Point<Integer>(10,10);
```

b

```
Point<Integer> iPoint = new Point<>(10,10);
```

c

```
Point<> iPoint = new Point<>(10,10);
```

d

```
Point<Integer> iPoint = new Point<Integer>(10,10);
```

```
float sum = 0;
for(int i=3; i>=0; i--){
    try{
        System.out.println(i);
        sum+= 1/i;
    }
    catch(NumberFormatException, ArrayIndexOutOfBoundsException exc) {
        System.out.println("NumberFormatException or ArrayIndexOutOfBoundsException");
    }
    catch(ArithmeticException exc) {
        System.out.println("ArithmeticException");
    }
}
System.out.println("Done");
```

a

3210ArithmeticeExceptionDone

b

321ArithmeticeExceptionDone

c

Program se neće izvesti zbog sintaksne pogreške, iznimke se ne učinjavaju sa zarezom nego sa znakom |

d

321NumberFormatException or ArrayIndexOutOfBoundsExceptionDone

e

321ExceptionDone

dogadja se to u jesenjoj noci kada pada kestenje po asfaltu i kada se cuju psi u daljini

Ako riječi dodajemo u TreeSet redoslijedom kojim su navedene u gornjoj rečenici, što ćemo dobiti kad sadržaj kolekcije ispišemo?

a dogadja se to u jesenjoj noci kada pada kestenje po asfaltu i cuju psi daljini

b asfaltu cuju daljini dogadja i jesenjoj kada kestenje noci pada po psi se to u

c se u noci to jesenjoj kestenje po cuju pada psi dogadja asfaltu daljini i kada

d asfaltu čuju daljini dogadja i jesenjoj kada kada kestenje noci pada po psi se se to u

Dokumentacija sučelja `java.util.Collection<T>` definira pojam optionalnih metoda. Što to znači za razrede koji implementiraju to sučelje i neće podržati akcije koje provode te metode?

- a** Takve metode razred obavezno mora označiti ključnom riječi `abstract`.
- b** Ti razredi trebaju imati implementirane takve metode na način da u tijelu istih izazovu iznimku `UnsupportedOperationException`.
- c** Ti razredi trebaju implementirati te metode ali ih označiti kao privatne (modifikator `private`) tako da ih korisnik ne može pozivati. Takve metode razred obavezno mora označiti ključnom riječi `abstract`.
- d** Implementacije tih metoda u razredu treba označiti posebnom anotacijom tako da kompjuter odbije prevesti kod koji bi ih pozivao.
- e** Ti razredi ne smiju definirati te metode tako da njihova implementacija nigdje ne postoji.

Što sve od navedenog je ispravno ako je *map* tipa `Map<String, List<Integer>>`?

a

```
for(Integer entry : map.values()) {  
    // radi nešto  
}
```

b

```
for(Map.Entry<String, Integer> entry : map.keySet()) {  
    // radi nešto  
}
```

c

```
for(Map.Entry<String, List<Integer>> entry : map.entrySet()) {  
    // radi nešto  
}
```

c

```
for(Map.Entry<String, List<Integer>> entry : map.entrySet()) {  
    // radi nešto  
}
```

d

```
for(List<Integer> entry : map.values()) {  
    // radi nešto  
}
```

e

```
for(Map.Entry<String, Integer> entry : map.entrySet()) {  
    // radi nešto  
}
```

Metode reverse, shuffle i sort dio su klase ili sučelja

a

List

b

Collection

c

Map

d

Set

e

Collections

Javni okvir kolekcija nudi sljedeće implementacije sučelja `java.util.List`:

a

ArrayList

b

MapList

c

HashList

d

LinkedList

e

TreeList

Što sve vrijedi za kolekciju tipa *List*

- a elementi nemaju svoju poziciju unutar liste
- b numeracija pozicija počinje od 1
- c može se dohvatiti element na zadanoj poziciji
- d elementi imaju svoju poziciju unutar liste
- e numeracija pozicija počinje od 0

šta od navedenog je ispravno ako je map tipa `Map<String, Integer>`?

a

```
for(String entry : map.keySet()) {  
    // radi nešto  
}
```

b

```
for(String entry : map.values()) {  
    // radi nešto  
}
```

c

```
for(var entry : map) {  
    // radi nešto  
}
```

d

```
for(String entry : map) {  
    // radi nešto  
}
```

e

```
for(String entry : map.entrySet()) {  
    // radi nešto  
}
```

Što od navedenog je ispravno ako je map tipa Map<String, Integer>?

a

```
for(Integer entry : map.entrySet()) {  
    // radi nešto  
}
```

b

```
for(var entry : map) {  
    // radi nešto  
}
```

c

```
for(Integer entry : map) {  
    // radi nešto  
}
```

d

```
for(Integer entry : map.values()) {  
    // radi nešto  
}
```

e

```
for(Integer entry : map.keySet()) {  
    // radi nešto  
}
```

Koju apstraktnu metodu je potrebno nadjačati kod implementacije sučelja `java.util.function.Predicat<T>`?

a

`boolean compare(T first, T second)`

b

`Predicate(boolean b)`

c

`int compareTo(T other)`

d

`void test(T t)`

e

`boolean test(T t)`

Implementacija sučelja `java.util.Set` iz Javinog okvira kolekcija koja osigurava da su elementi sortirani je:

a HashSet

b LinkedHashSet

c TreeSet

d ComparableSet

Implementacija sučelja `java.util.Set` iz Javineg okvira kolekcija koja osigurava da su elementi poredani redoslijedom dodavanja je:

a `OrderedSet`

b `TreeSet`

c `LinkedHashSet`

d `HashSet`

dogadja se to u jesenjoj noci kada pada kestenje po asfaltu i kada se cuju psi u daljini

Ako riječi dodajemo u TreeSet redoslijedom kojim su navedene u gornjoj rečenici, što ćemo dobiti kad sadržaj kolekcije ispišemo?

a se u noci to jesenjoj kestenje po cuju pada psi dogadja asfaltu daljini i kada

b dogadja se to u jesenjoj noci kada pada kestenje po asfaltu i cuju psi daljini

c asfaltu cuju daljini dogadja i jesenjoj kada kestenje noci pada po psi se to u

d asfaltu čuju daljini dogadja i jesenjoj kada kada kestenje noci pada po psi se se to u u

Dokumentacija sučelja `java.util.Collection` traži od razreda koji implementiraju to sučelje (ili neko iz njega izvedeno) da bi trebali podržati određen broj konstruktora. Od ponuđenih, o kojima se radi?

- a konstruktor koji prima referencu na jedan `Iterator` te u konstruiranu kolekciju dodaje sve elemente koje iterator vrati
- b konstruktor koji ne prima niti jedan argument i stvara praznu kolekciju
- c konstruktor koji prima referencu na jedan `Iterable` te u konstruiranu kolekciju dodaje sve elemente koje vrati stvoreni iterator
- d konstruktor koji prima referencu na neku drugu kolekciju te u konstruiranu kolekciju kopira sve elemente iz predane kolekcije
- e konstruktor koji prima referencu na dvije kolekcije te u konstruiranu kolekciju kopira sve elemente najprije iz prve a potom iz druge kolekcije

b

Program će baciti iznimku *DuplicateKeyException*

c

2

4, 5

d

1

5

e

2

5, 4

Ako je mapa definirana s `Map<String, Integer> map` ispravno inicijalizirana te je prazna u trenutku izvođenja sljedećih naredbi,

```
map.put("Ana", 5);
map.put("Ana", 4);
System.out.println(map.size());
System.out.println(map.get("Ana"));
```

što će se ispisati na ekranu?

a

1

4

b

Program će baciti iznimku *DuplicateKeyException*

c

2

4,5

Metoda vrtiKolekciju() vraća ne-null referencu na neku nepromjenjivu kolekciju. Što je ponuđenoga potrebno upisati na mjesto označeno podvukama u kodu kako met ne bi svojem pozivatelju propagirala iznimku nastalu uslijed poziva metode add u bloku try?

```
public void m() {  
    try {  
        Collection<String> kolekcija = vrtiKolekciju();  
        kolekcija.add("Pero");  
        System.out.println(kolekcija.contains("Pero"));  
    }  
    catch(_____) { }  
}
```

a

NullPointerException ex

b

UnsupportedOperationException ex

c

IndexOutOfBoundsException ex

a

NullPointerException ex

b

UnsupportedOperationException ex

c

IndexOutOfBoundsException ex

d

IllegalArgumentException ex

Ako je skup definiran s `Set<String> set` ispravno inicijaliziran te je prazan u trenutku izvođenja sljedećih naredbi,

```
System.out.println(set.add("Ana"));
System.out.println(set.add("Ana"));
```

što će se ispisati na ekranu?

a

false

Ana

b

1

2

c

true

false

Koja izjava vrijedi za `java.util.LinkedList`:

- a** lista za pohranu elemenata koristi dinamičko polje te se polje povećava po potrebi
- b** lista elemente pohranjuje dinamički alocirajući nove čvorove za dvostruko povezanu listu

Što od navedenog je istina za `java.util.Collections`?

a

`java.util.Collections` sadrži samo statičke metode

b

To je klasa

c

To je statička metoda

d

To je sučelje

e

To je metoda

Koju apstraktну методу је потребно надјаћати код реализације суčelja `java.util.function.Predicate<T>`?

a

`int compareTo(T other)`

b

`Predicate(boolean b)`

c

`void test(T t)`

d

`boolean compare(T first, T second)`

e

`boolean test(T t)`

Listu možemo stvoriti pozivom:

a

`Arrays.asList(1,2,3);`

b

`Arrays.asList("A", "B", "C");`

c

`new List();`

d

`new LinkedList();`

e

`new ArrayList<>();`

Koja izjava vrijedi za `java.util.LinkedList`:

- a lista elemenata pohranjuje dinamički alocirajući nove čvorove za dvostruko povezanu listu
- b lista za pohranu elemenata koristi dinamičko polje te se polje povećava po potrebi

Ako je skup definiran s `Set<String> set` ispravno inicijaliziran te je prazan u trenutku izvođenja sljedećih naredbi,

```
System.out.println(set.add("Ana"));
System.out.println(set.add("Ana"));
```

što će se ispisati na ekranu?

a

true

false

b

false

Ana

c

1

1

d

1

2

e

false

true

Što od navedenog je ispravno ako je *map* tipa `Map<String, Integer>`?

a

```
for(Map.Entry<String, Integer> entry : map.values()) {  
    // radi nešto  
}
```

b

```
for(Map.Entry<String, Integer> entry : map.keySet()) {  
    // radi nešto  
}
```

c

```
for(Map.Entry<String, Integer> entry : map.entrySet()) {  
    // radi nešto  
}
```

d

```
for(var entry : map) {  
    // radi nešto  
}
```

e

```
for(Map.Entry<String, Integer> entry : map) {  
    // radi nešto  
}
```

Metode reverse, shuffle i sort dio su klase ili sučelja

a

Collections

b

List

c

Collection

d

Map

e

Set

Koja izjava vrijedi za `java.util.LinkedList`:

a

lista za pohranu elemenata koristi dinamičko polje te se polje povećava po potrebi

b

lista elemente pohranjuje dinamički alocirajući nove čvorove za dvostruko povezanu listu

Most između znakovnih tokova i tokova okteta je

a

OutputStreamWriter

b

FileWriter

c

StringWriter

d

WritableStringStream

Ako je students definiran kao List<Student>, kojeg je

tipa rez ako je

```
rez = students.stream().filter( s -> s.getFinalGrade()>3).map(s -> .getLastName()).collect(Collectors.toList())
```

a Optional<Student>

b Optional<String>

c Stream<Student>

d List<String>

e Stream<String>

Za pozadinske poslove prilikom obrade grafičkih događa koristi se klasa SwingWorker. Koju od navedenih metoda moramo implementirati?

a get

b publish

c process

d doInBackground

e done

Pretpostavimo da imamo mapu mapa tipa `Map<String, Integer>`. Koji će se od ponuđenih načina uspješno prevesti i ispisati sve vrijednosti pridružene ključevima?

a

```
mapa.forEach(System.out.println(value));
```

b

```
mapa.forEach((k,v) -> System.out.println(v));
```

c

```
mapa.forEach((k,v) -> { System.out.println(v); });
```

d

```
mapa.forEach(System.out::println);
```

e

```
mapa.forEach(System.out.println);
```

Koje od sljedećih tvrdnji vrijede za anonimne klase?

- a** nemaju ime i najčešće se koriste kad nam je dovoljno stvoriti samo jedan primjerak
- b** definicija anonimne klase se piše na mjestu stvaranja primjerka u vitičastim zagradama
- c** moraju imati eksplizitni konstruktor
- d** moraju se definirati samostalno tj. ne mogu biti klase koje implementiraju neko sučelje ili nasljeđuju od neke druge klase

Paket `java.io` podržava sljedeće vrste tokova podataka:

a

tokovi cijelih brojeva (int)

b

tokovi datoteka (file)

c

tokovi znakova (char)

d

tokovi okteta (byte)

Koje su tipične naredbe za početak metode equals koja je nadjačana u klasi Student?

a

```
@Override  
public boolean equals(Object obj) {  
    if (obj == null) return false;  
    if (!(obj instanceof Student))  
        return false;  
    // ...
```

b

```
@Override  
public boolean equals(Object obj) {  
    if (obj == null) return false;  
    if (!(obj instanceof Student))  
        return true;  
    // ...
```

Ako imamo definiran razred `CompositeComparator<T>` koji uspoređuje redom po predanim komparatorima te razred `Car` koji ima definiran prirodni komparator (uspoređuje id automobila) i tri članske varijable koje predstavljaju primitivne komparatore po članskim varijablama (brand, model i color), što će ispisati sljedeći programski odsječak?

```
Comparator<Car> comparator = new CompositeComparator<>(
    Car.BY_BRAND.reversed(), Car.BY_MODEL, Car.BY_COLOR.reversed(), Comparator.naturalOrder());
Set<Car> cars = new TreeSet<>(comparator);
Car car1 = new Car("Fiat", "Punto", "Blue", "6");
Car car2 = new Car("BMW", "M3", "Black", "2");
Car car3 = new Car("Mercedes", "AMG63", "Silver", "3");
Car car4 = new Car("Fiat", "Punto", "Blue", "4");
cars.add(car1);
cars.add(car2);
cars.add(car3);
cars.add(car4);
for (Car c : cars) {
    System.out.print(c.brand + "_" + c.model + "_" + c.color + "_" + c.id + " ");
}
```

a

Mercedes_AMG63_Silver_3 Fiat_Punto_Blue_4 Fiat_Punto_Blue_6 BMW_M3_Black_2

b

BMW_M3_Black_2 Mercedes_AMG63_Silver_3 Fiat_Punto_Blue_4 Fiat_Punto_Blue_6

c

Fiat_Punto_Blue_4 Fiat_Punto_Blue_6 Mercedes_AMG63_Silver_3 BMW_M3_Black_2

d

Fiat_Punto_Blue_4 BMW_M3_Black_2 Mercedes_AMG63_Silver_3 Fiat_Punto_Blue_6

a

izvedena klasa iz tipa koji je definiran u toj metodi

b

tip definiran u toj metodi

c

bilo koji tip

d

sve od navedenog

```
interface I1 { }
class C1 implements I1 { }
public class Main {
    public static void main(String[] args) {
        I1 a = new C1(); // (1)
        C1 b = new C1(); // (2)
        b = a; // (3)
        a = b; // (4)
        a = new I1(); // (5)
    }
}
```

a

C1 b = new C1(); // (2)

b

b = a; // (3)

c

a = b; // (4)

d

I1 a = new C1(); // (1)

e

a = new I1(); // (5)

U kojoj liniji koda će doći do greške tijekom izvođenje sljedećeg programskog odsječka, ako metoda `StudentData.load` vraća listu studenata.

```
List<Student> students = StudentData.load();           //1
Stream<Student> st = students.stream();
st.forEach(t -> System.out.println(t));
st.forEach(t -> System.out.println(t));               //2
students.stream().forEach(t -> System.out.println(t)); //3
```

a 3

b 2

c 1

d 5

e 4

po tipovima parametara koje prihvaca. Neka razred R3 nasljeđuje razred R2, a razred R2 nasljeđuje razred R1. Prepostavimo da imamo mapu mapa tipa Map<Number> koristi kao argument u pozivu mapa.forEach(____). Koji sve odgovori predstavljaju ispravno napisan kod (u smislu da će se uspješno prevesti)?

Za model događaja u grafičkom sučelju napravljenom korištenjem Swinga vrijedi:

- a** nakon okidanja događaja pozivaju se metode u svakom registriranom promatraču
- b** jedan ili više promatrača mogu biti pretplaćeni na događaje
- c** komponente okidaju događaje
- d** promatrači određenog događaja moraju implementirati specifično sučelje za obradu tog događaja
- e** u metodi za obradu specifičnog događaja se proslijeđuje objekt koji u sebi ima referencu na objekt koji je izvor tog događaja

Koje su od sljedećih tvrdnji vezanih za sučelje `Predicate<T>` u Javi točne?

a

sučelje `Predicate` je funkcionalno sučelje čija je apstraktna metoda boolean `test(T t)`

b

sučelje `Predicate` je funkcionalno sučelje čija je apstraktna metoda void `accept(T t)`

c

metoda `test` služi da bi se provjerilo vrijedi li neki uvjet za dani objekt

Mostovi između znakovnih tokova i tokova okteta su

a

FileReader

b

InputStreamReader

c

OutputStreamWriter

d

FileWriter

Koji od ponuđenih odgovora će u mapi `Map<String, Integer>` Ivani povećati broj bodova za 2 (ili upisati 2 ako ništa još nije upisano)? Podsjetnik: metoda `Map.merge` redom dobiva ključ, početnu vrijednost te bifunkciju. Napomena: razred `Integer` ima statičku metodu `int sum(int a, int b)`.

a

```
mapa.merge("Ivana", 2, vo,vi -> vo+vi);
```

b

```
mapa.merge("Ivana", 2, (vo,vi) -> return vo+vi);
```

c

```
mapa.merge("Ivana", 2, (vo,vi) -> Integer.sum(vo,vi));
```

d

```
mapa.merge("Ivana", 2, Integer::sum);
```

e

```
mapa.merge("Ivana", 2, Integer.sum(a,b));
```

f

```
mapa.merge("Ivana", 2, (vo,vi) -> vo+vi);
```

eka klasa Student ne implementira nijedno sučelje niti je eksplisitno izvedena iz neke druge klase te neka je definirana klasa StudentComparator takva da implementira `comparator<Student>`.

ko su `s1` i `s2` tipa `Student`, `c1` i `c2` tipa `StudentComparator`, što od navedenog je (sintaksno) ispravno?

a

`s1.compareTo(s2, c1)`

b

`c1.compareTo(c2)`

c

`c1.compare(s1, s2)`

d

`Comparator(c1).compare(s1, s2)`

e

`Student.compareTo(s1, s2, c1)`

mo definirali ako u nekom *JFrame*u postavio layout na `new GridLayout(3, 0)`

A Za razmještaj komponenti koristi se tablica koja ima 2 ili manje redaka, ovisno o broju dodanih komponenti

B Za razmještaj komponenti koristi se tablica koja barem 3 retka, ali može i više ovisno o broju dodanih komponenti

C Za razmještaj komponenti koristi se tablica koja ima točno 3 retka, a broj stupaca se određuje prema broju redaka i broju dodanih komponen

D u taj *JFrame* se mogu dodati maksimalno 3 komponente

E Ništa jer je *JFrame* krovni kontejner i ne može mu se definirati razmještaj elemenata.

Označite ispravne načine ispis svih studenata iz liste `List<Student> students` :

a

```
students.stream().forEach(t -> System.out.println(t));
```

b

```
students.stream().forEach(System.out.println(t));
```

c

```
students.stream().forEach().println();
```

d

```
students.stream().forEach(new Consumer<Student>() {  
    @Override  
    public void accept(Student t) {  
        System.out.println(t);  
    }});
```

e

```
students.stream().forEach().println(Student t);
```

implementiramo iterator pomoću statičke ugnježđene klase (MyIterator) koja se nalazi u klasi MyCollection onda za tu klasu vrijedi:

a

kroz konstruktor takve klase moramo poslati referencu na vanjsku klasu te se takva referenca mora spremiti u atribut

b

klasu možemo instancirati sa sljedećim kodom

```
new MyCollection.MyIterator.iterator()
```

c

instanci vanjske klase možemo pristupiti na sljedeći način

```
MyCollection.this
```

d

klasu možemo instancirati sa sljedećim kodom

```
new MyCollection.MyIterator(...)
```

e

klasu možemo instancirati sljedećim kodom

```
MyCollection collection = new MyCollection(); MyIterator iterator = collection.new MyIterator(...);
```

Metoda forEach definirana u sučelju `Map<K, V>` prihvata referencu na objekt tipa BiConsumer koji je maksimalno fleksibilan po tipovima parametara koje prihvata. Neka metoda stvori() stvara i vraća objekt koji je BiConsumer identičan tipu s lijeve strane izraza, nakon čega se taj objekta koristi kao argument u pozivu mapu.

a

```
BiConsumer<Object, R1> a = stvori(); mapa.forEach(a);
```

b

```
BiConsumer<Number, R3> a = stvori(); mapa.forEach(a);
```

c

```
BiConsumer<Integer, R2> a = stvori(); mapa.forEach(a);
```

d

```
BiConsumer<Integer, R1> a = stvori(); mapa.forEach(a);
```

e

```
BiConsumer<Integer, R3> a = stvori(); mapa.forEach(a);
```

f

```
BiConsumer<Number, R2> a = stvori(); mapa.forEach(a);
```