

Čo sa stane ak sa pokúsime skompilovať a spustiť nasledovný kód? (Poznámka: Number je predok Integer.)

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List<? super Integer> sList = new ArrayList<Number>();  
4.         int i = 2007;  
5.         sList.add(i);  
6.         sList.add(++i);  
7.         Number num = sList.get(0);  
8.     }  
9. }
```

Select one or more:

- a. Kód sa neskompiluje kvôli chybe na riadku 3.
- b. Kód sa neskompiluje kvôli chybe na riadku 6.
- c. Kód sa skompiluje, ale po spustení program vyhodí výnimku ClassCastException na riadku 7.
- d. Kód sa neskompiluje kvôli chybe na riadku 7.
- e. Kód sa skompiluje a spustí normálne.

Majme nasledovné deklarácie:

```
interface A {}  
class B {}  
class C extends B implements A {}  
class D implements A {}  
B b = new B();  
C c = new C();  
D d = new D();
```

Ktoré z nasledovných priradení sú správne?

Označte jednu alebo viac odpovedí:

- a. c = b;
- b. d = (D) c;
- c. d = c;
- d. A a = d;
- e. c = d;

Ktoré z deklarácií členskej premennej accounts je možné použiť na riadku 2 uvedeného kódu?

```
1. public class Testing {  
2.     // VLOŽTE SPRÁVNU DEKLARÁCIU ČLENSKEJ PREMENNEJ  
3.     public long getNum(String name) {  
4.         Long number = accounts.get(name);  
5.         return number == null ? 0 : number;  
6.     }  
7.     public void setNum(String name, long number) {  
8.         accounts.put(name, number);  
9.     }  
10. }
```

Označte jednu alebo viac odpovedí:

- a. private Map accounts = new HashMap();
- b. private Map<String, long> accounts = new HashMap<String, long>();
- c. private Map<String, Long> accounts = new HashMap<String, Long>();
- d. private Map<String<Long>> accounts = new HashMap<String<Long>>();
- e. private Map<String, Long> accounts = new Map<String, Long>();

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

```
1. class Main {  
2.     public static void main(String[] args) {  
3.         System.out.println("Hello");  
4.     }  
5. }
```

Trieda GenericFruit deklaruje nasledovnú metódu:

```
public void setCalorieContent(float f)
```

Pri vytváraní triedy Apple odvodenej z triedy GenericFruit potrebujeme pridať metódy, ktoré preťažujú (overload) túto metódu. Ktoré z nasledovných deklarácií metód sú pre tento prípad použiteľné?

Select one or more:

- a. protected void setCalorieContent(float x)
- b. protected float setCalorieContent(String s)
- c. public void setCalorieContent(String s) throws NumberFormatException
- d. public void setCalorieContent(double d)

Aký bude výsledok po skompilovaní a spustení nasledovného kódu?

```
1. class Fruit {}  
2. class Apple extends Fruit {}  
3. class Orange extends Fruit {}  
4. public class Testing {  
5.     public static void main(String[] args) {  
6.         ArrayList<Apple> aList = new ArrayList<Apple>();  
7.         aList.add(new Apple());  
8.         ArrayList<Orange> oList = bList;  
9.         oList.add(new Orange());  
10.        System.out.println(aList);  
11.    }  
12. }  
13. }
```

Select one or more:

- a. Kód sa skompiluje bez chýb a varovaní a po spustení vypíše text "[Apple@hhhhh, Orange@HHHHHHH]", pričom hhhhh a HHHHHHH reprezentujú nejaký hash kód.
- b. Kód sa neskompiluje kvôli chybe na riadku 8.
- c. Kód sa skompiluje, ale ohlásí neošetrenú konverziu na riadku 9.
- d. Kód sa skompiluje, ale ohlásí neošetrenú konverziu na riadku 8.
- e. Kód sa skompiluje a po spustení hodí výnimku ClassCastException na riadku 9.

Čo bude na výstupe po spustení nasledovného programu?

```
1. public class MyClass {  
2.     public static void main(String[] args) {  
3.         try {  
4.             f();  
5.         } catch (InterruptedException e) {  
6.             System.out.println("1");  
7.             throw new RuntimeException();  
8.         } catch (RuntimeException e) {  
9.             System.out.println("2");  
10.            return;  
11.         } catch (Exception e) {  
12.             System.out.println("3");  
13.         } finally {  
14.             System.out.println("4");  
15.         }  
16.         System.out.println("5");  
17.     }  
18.     // InterruptedException je priamy potomok Exception  
19.     static void f() throws InterruptedException {  
20.         throw new InterruptedException("Time for lunch.");  
21.     }  
22. }
```

Select one or more:

- a. Program vypíše "1" a "4".
- b. Program vypíše "1", "4" a "5".
- c. Program vypíše "1", "2", "4" a "5".
- d. Program vypíše "3" a "5".
- e. Program vypíše "1", "2" a "4".
- f. Program vypíše "5".

Question 6Not yet
answeredMarked out of
3.00

Flag question

Ktoré z deklarácií členskej premennej accounts je možné použiť na riadku 2 uvedeného kódu?

```
1. public class Testing {  
2.     // VLOŽTE SPRÁVNU DEKLARÁCIU ČLENSKEJ PREMENNEJ  
3.     public long getNum(String name) {  
4.         Long number = accounts.get(name);  
5.         return number == null ? 0 : number;  
6.     }  
7.     public void setNum(String name, long number) {  
8.         accounts.put(name, number);  
9.     }  
10. }
```

Select one or more:

- a. private Map accounts = new HashMap();
- b. private Map<String, long> accounts = new HashMap<String, long>();
- c. private Map<String, Long> accounts = new HashMap<String, Long>();
- d. private Map<String<Long>> accounts = new HashMap<String<Long>>();
- e. private Map<String, Long> accounts = new Map<String, Long>();

Ktoré z nasledovných deklarácií sa skomplilujú bez varovných správ?

Select one or more:

- a. `Map<? super Integer, ? super Integer> map5 = new HashMap<Number, Number>();`
- b. `Map<Integer, Map<Integer, String>> map1 = new HashMap<Integer, HashMap<Integer, String>>();`
- c. `Map<Integer, Integer> map3 = new HashMap<Integer, Integer>();`
- d. `Map<Integer, HashMap<Integer, String>> map2 = new HashMap<Integer, HashMap<Integer, String>>();`
- e. `Map<?, ?> map7 = new HashMap<?, ?>();`
- f. `Map<? extends Number, ? extends Number> map6 = new HashMap<Number, Number>();`
- g. `Map<? super Integer, ? super Integer> map4 = new HashMap<? super Integer, ? super Integer>();`

Majme nasledovný kód:

```
1. public class Exceptions {  
2.     public static void main(String[] args) {  
3.         try {  
4.             if (args.length == 0) return;  
5.             System.out.println(args[0]);  
6.         } finally {  
7.             System.out.println("The end");  
8.         }  
9.     }  
10. }
```

Ktoré z tvrdení sú pravdivé?

Select one or more:

- a. Ak spustíme program bez argumentov, program nebude mať žiadnen výstup.
- b. Ak spustíme program bez argumentov, program vypíše "The end".
- c. Ak spustíme program s jedným argumentom, program vypíše tento argument.
- d. Ak spustíme program s jedným argumentom, program vypíše tento argument a za ním vypíše "The end".
- e. Program vyhodí výnimku `ArrayIndexOutOfBoundsException`.

```
1. class A {  
2.     void f() throws ArithmeticException { }  
3. }  
4. public class MyClass extends A {  
5.     public static void main(String[] args) {  
6.         A obj = new MyClass();  
7.         try {  
8.             obj.f();  
9.         } catch (ArithmeticException e) {  
10.             return;  
11.         } catch (Exception e) {  
12.             System.out.println(e);  
13.             throw new RuntimeException("Something wrong here");  
14.         }  
15.     }  
16.     // InterruptedException je priamy potomok Exception.  
17.     void f() throws InterruptedException { }  
18. }
```

Select one or more:

- a. Kompilátor bude namietať, že blok catch (ArithmeticException) zatieňuje blok catch (Exception).
- b. Nemôžeme vyhadzovať výnimku v bloku catch.
- c. Prekrytá metóda f() v MyClass nemôže vyhadzovať InterruptedException, keďže v A takúto výnimku nevyhadzuje.
- d. Prekrytá metóda f() v MyClass musí deklarovať, že vyhadzuje ArithmeticException, keďže je to tak v triede A.
- e. Všetko je v poriadku, komplilácia prebehne bez chýb.
- f. Metóda main() musí deklarovať, že vyhadzuje RuntimeException.

Question 9Not yet
answeredMarked out of
3.00

Flag question

Ktorú z deklarácií metódy justDoIt() môžeme použiť na riadku 9, aby sme nedostali varovné správy pri kompliacii?

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List raw = new ArrayList();  
4.         raw.add("2007");  
5.         raw.add(2008);  
6.         raw.add("2009");  
7.         justDoIt(raw);  
8.     }  
9.     // SEM VLOŽTE SPRÁVNU DEKLARÁCIU FUNKCIE  
10. }
```

Select one or more:

- a. static void justDoIt(List<T> lst) { }
- b. Žiadna z uvedených deklarácií.
- c. static void justDoIt(List<Integer> lst) { }
- d. static void justDoIt(List<?> lst) { }

Question 10Not yet
answeredMarked out of
3.00

Flag question

Ktoré z nasledovných deklarácií tried sú správne a deklarujú triedu, z ktorej sa nedajú vytvárať objekty?

Select one or more:

- a. abstract class Ghost { void haunt(); }
- b. static class Ghost { abstract haunt(); }
- c. abstract class Ghost { void haunt() {} }
- d. class Ghost { abstract void haunt(); }
- e. abstract Ghost { abstract void haunt(); }

Question 7

Not yet
answered

Marked out of
3.00

Flag question

Ktoré z príkazov, keď ich vložíme na riadok 4, nespôsobia chybu kompliacie?

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List<?> lst = new ArrayList<String>();  
4.         // SEM VLOŽTE SPRÁVNY PRÍKAZ  
5.     }  
6. }
```

Select one or more:

- a. lst.add(null);
- b. Object v2 = lst.get(0);
- c. lst.add("OK");
- d. String v1 = lst.get(0);
- e. lst.add(2007);

Time left 0:07:01

Question 3Not yet
answeredMarked out of
3.00

Flag question

Majme definovanú nasledovnú triedu:

```
1. class Widget extends Thingee {  
2.     static final int maxWidgetSize = 40;  
3.     static String title;  
4.     public Widget(int mx, String t) {  
5.         maxWidgetSize = mx;  
6.         title = t;  
7.     }  
8.     // ďalší kód  
9. }
```

Čo sa stane, keď sa pokúsime skompilovať tento kód a spustiť program, ktorý vytvára objekt Widget nasledovným spôsobom?

```
10. Widget myWidget = new Widget(50, "Bigger");
```

Select one or more:

- a. Program sa skompliluje, ale počas behu programu nastane chyba na riadku 5.
- b. Program sa skompliluje a spustí bez chyby.
- c. Nastane chyba pri komplilácii na riadku 6.
- d. Nastane chyba pri komplilácii na riadku 5.

Question 4Not yet
answeredMarked out of
3.00

Flag question

Ktorú z deklarácií metódy justDoIt() môžeme použiť na riadku 9, aby sme nedostali varovné správy pri komplilácii?

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List raw = new ArrayList();  
4.         raw.add("2007");  
5.         raw.add(2008);  
6.         raw.add("2009");  
7.         justDoIt(raw);
```

БУХО ↑



1°C

Cloudy

13:08
17.01.2025

Ktoré konštruktory môžeme vložiť na riadok 11, aby nedošlo ku chybe pri kompliacii?

```
1. class MySuper {  
2.     int number;  
3.     MySuper(int i) { number = i; }  
4. }  
5. class MySub extends MySuper {  
6.     int count;  
7.     MySub(int count, int num) {  
8.         super(num);  
9.         this.count = count;  
10.    }  
11.    // SEM VLOŽTE SPRÁVNY KONŠTRUKTOR  
12. }
```

Select one or more:

- a. MySub(int count) { this.count = count; super(count); }
- b. MySub(int count) { this.count = count; }
- c. MySub(int count) { super(count); this(count, 0); }
- d. MySub(int count) { super(); this.count = count; }
- e. MySub() {}
- f. MySub(int count) { this(count, count); }

- c. Kompilátor odmietne novú funkciu, pretože má iný modifikátor prístupu.
- d. Táto technika sa nazýva overloading (preťaženie).

Time left 0:03:33

Question 6Not yet
answeredMarked out of
3.00

Flag question

Ked' sa pokúsite skompilovať nasledovný kód, objaví sa správa, že premenná tmp nie je inicializovaná:

```
1. class Demo {  
2.     String msg = "Type is ";  
3.     public void showType(int n) {  
4.         String tmp;  
5.         if (n > 0) tmp = "positive";  
6.         System.out.println(msg + tmp);  
7.     }  
8. }
```

Ktoré z nasledujúcich úprav umožnia eliminovať túto správu komplilátora?

Select one or more:

- a. Vložiť nový riadok medzi riadky 5 a 6: else tmp = "not positive";
- b. Presunúť riadok 4 a vložiť ho medzi riadky 2 a 3 (čím sa tmp stane členskou premennou).
- c. Zmeniť riadok 4 na: String tmp = "";
- d. Zmeniť riadok 4 na: String tmp = null;

Question 7Not yet
answeredMarked out of
3.00

Aký bude výsledok po skomplilovaní a spustení nasledovného programu?

```
1. class Vehicle {}  
2. class Car extends Vehicle {}  
3. class Sedan extends Car {}  
4. class Coupe extends Car {}
```

Question 4

Not yet
answered

Marked out of
3.00

Flag question

Pracuješ na triede `Aquarius`, ktorá bude simulovať akvárium. Už máš metódu, ktorá pridá rybu do akvária a vráti zostávajúcu voľnú kapacitu akvária. Táto metóda ma nasledovnú deklaráciu:

```
public int addFish(Fish f)
```

Teraz chceš vytvoriť metódu na vloženie celej skupiny rýb naraz. Navrhovaná metóda má nasledovnú deklaráciu:

```
protected boolean addFish(Fish[] f)
```

Myšlienka je, že metóda vráti `true`, ak je ešte miesto v akváriu, alebo `false`, ak už tam viac miesta nie je. Ktoré z nasledujúcich tvrdení sú pravdivé?

Select one or more:

- a. Kompilátor odmietne novú funkciu, pretože má iný modifikátor prístupu.
- b. Táto technika sa nazýva overloading (preťaženie).
- c. Kompilátor odmietne novú funkciu pretože má iný návratový typ.
- d. Táto technika sa nazýva ~~overriding~~ (prekrývanie).

Aký bude výsledok po pokuse skompilovať a spustiť nasledovný kód? (Poznámka: Number je predok Integer.)

```
1. public class TestList {  
2.     public static void main(String[] args) {  
3.         List<Integer> lst = new ArrayList<Integer>();  
4.         lst.add(2007);  
5.         lst.add(2008);  
6.         List<Number> numList = lst;  
7.         for (Number n : numList)  
8.             System.out.println(n + " ");  
9.     }  
10. }
```

Select one or more:

- a. Nastane chyba pri komplácii na riadku 3.
- b. Nastane chyba pri komplácii na riadku 7.
- c. Program sa skompliluje a po spustení vypíše "2007 2008".
- d. Program sa skompliluje, ale počas vykonávania sa vyhodí výnimka ClassCastException na riadku 6.
- e. Nastane chyba pri komplácii na riadku 6.

Čo vypíše nasledovný program po jeho spustení?

```
1. public class MyClass {  
2.     public static void main(String[] args) {  
3.         int k = 0;  
4.         try {  
5.             int i = 5 / k;  
6.         } catch (ArithmetricException e) {  
7.             System.out.println("1");  
8.         } catch (Runtimeexception e) {  
9.             System.out.println("2");  
10.        return;  
11.    } catch (Exception e) {  
12.        System.out.println("3");  
13.    } finally {  
14.        System.out.println("4");  
15.    }  
16.    System.out.println("5");  
17. }  
18. }
```

Označte jednu alebo viac odpovedí:

- a. Program vypíše "1" a "4".
- b. Program vypíše "1", "2" a "4".
- c. Program vypíše "1", "4" a "5".
- d. Program vypíše "3" a "5".
- e. Program vypíše "5".
- f. Program vypíše "1", "2", "4" a "5".

AI KAXE, ALE X

3

Ktoré z tvrdení sú pravdivé pre nasledovný kód?

```
1. class Fruit {}  
2. class Apple extends Fruit {}  
3. public class TestApp {  
4.     public static void main(String[] args) {  
5.         List<? extends Apple> lst1 = new ArrayList<Fruit>();  
6.         List<? extends Fruit> lst2 = new ArrayList<Apple>();  
7.         List<? super Apple> lst3 = new ArrayList<Fruit>();  
8.         List<? super Fruit> lst4 = new ArrayList<Apple>();  
9.         List<?> lst5 = lst1;  
10.        List<?> lst6 = lst3;  
11.        List lst7 = lst6;  
12.        List<?> lst8 = lst7;  
13.    }  
14. }
```

Select one or more:

- a. Ani jedna odpoveď nie je správna.
- b. Riadok 11 sa skompliluje, riadok 12 nie.
- c. Riadok 7 sa skompliluje, riadok 8 nie.
- d. Riadok 5 sa skompliluje, riadok 6 nie.
- e. Riadok 9 sa skompliluje, riadok 10 nie.

Majme nasledovné deklarácie rozhraní, tried a referencií:

```
interface I1 {}  
interface I2 {}  
class C1 implements I1 {}  
class C2 implements I2 {}  
class C3 extends C1 implements I2 {}  
C1 obj1;  
C2 obj2;  
C3 obj3;
```

Vyberte, ktoré z nasledovných priradení sú správne:

Select one or more:

- a. I1 a = obj2;
- b. I1 b = obj3;
- c. I2 c = obj1;
- d. obj3 = obj1;
- e. obj3 = obj2;
- f. obj2 = obj1;

Kde je chyba v nasledovnom kóde?

```
1. public class MyClass {  
2.     public static void main(String[] args) throws A {  
3.         try {  
4.             f();  
5.         } finally {  
6.             System.out.println("Done.");  
7.         } catch (A e) {  
8.             throw e;  
9.         }  
10.    }  
11.    public static void f() throws B {  
12.        throw new B();  
13.    }  
14.}  
15. class A extends Throwable {}  
16. class B extends A {}
```

Select one or more:

- a. V metóde main() musí blok finally nasledovať za blokom catch.
- b. Metóda main() musí deklarovať, že vyhadzuje výnimku typu B.
- c. Za blok try nesmie nasledovať zároveň obidva bloky finally aj catch.
- d. Deklarácia triedy A nie je správna.
- e. V metóde main() musí catch blok deklarovať, že odchytáva výnimky typu B a nie typu A.

Question 8Not yet
answeredMarked out of
3.00

Flag question

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

```
1. class Vehicle {  
2.     static public String getModelName() { return "Volvo"; }  
3.     public long getRegNo() { return 12345; }  
4. }  
5. class Car extends Vehicle {  
6.     static public String getModelName() { return "Toyota"; }  
7.     public long getRegNo() { return 54321; }  
8. }  
9. public class TakeARide {  
10.    public static void main(String args[]) {  
11.        Car c = new Car();  
12.        Vehicle v = c;  
13.        System.out.println("|" + v.getModelName() + "|" + c.getModelName() +  
14.                           "|" + v.getRegNo() + "|" + c.getRegNo() + "|");  
15.    }  
16. }
```

Select one or more:

- a. Kód sa skompiluje a po spustení vypíše "|Volvo|Volvo|12345|54321|".
- b. Kód sa skompiluje a po spustení vypíše "|Toyota|Volvo|12345|54321|".
- c. Kód sa skompiluje a po spustení vypíše "|Toyota|Volvo|12345|12345|".
- d. Program sa nebude dať skompilovať.
- e. Kód sa skompiluje a po spustení vypíše "|Volvo|Toyota|54321|54321|".
- f. Kód sa skompiluje a po spustení vypíše "|Toyota|Toyota|12345|12345|".
- g. Kód sa skompiluje a po spustení vypíše "|Volvo|Toyota|12345|54321|".

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

```
1. class Vehicle { }
2. class Car extends Vehicle { }
3. class Sedan extends Car { }
4. class Garage<V> {
5.     private V v;
6.     public V get() { return this.v; }
7.     public void put(V v) { this.v = v; }
8. }
9. public class GarageAdmin {
10.    private Object object = new Object();
11.    private Vehicle vehicle = new Vehicle();
12.    private Car car = new Car();
13.    private Sedan sedan = new Sedan();
14.    public void doD(Garage<? extends Car> g) {
15.        g.put(object);
16.        g.put(vehicle);
17.        g.put(car);
18.        g.put(sedan);
19.        object = g.get();
20.        vehicle = g.get();
21.        car = g.get();
22.        sedan = g.get();
23.    }
24. }
```

Označte jednu alebo viac odpovedí:

- a. Priradenia na riadkoch 19, 20 a 21 sa skompilujú.
- b. Volanie metódy `put()` na riadkoch 15 a 16 sa skompilujú.
- c. Priradenie na riadku 22 sa skompiluje.
- d. Volanie metódy `put()` na riadkoch 17 a 18 sa skompilujú.

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

```
1. class Vehicle {
2.     static public String getModelName() { return "volvo"; }
3. }
```

Majme vlastnú hierarchiu výnimiek odvodenú od `java.lang.Exception` nasledovne (BitterException a SourException sú potomkami BadTasteException):

```
Exception
  +-- BadTasteException
    --- BitterException
    --- SourException
```

Vaša trieda, napr. `BaseCook`, má metódu deklarovanú nasledovne:

```
int rateFlavor(Ingredient[] list) throws BadTasteException
```

Trieda TexMexCook, zdedená z `BaseCook` ma metódu prekryvajúcu `rateFlavor()`. Ktoré z nasledovných deklarácií prekrytej metódy sú správne?

Select one or more:

- a. int rateFlavor(Ingredient[] list)
- b. int rateFlavor(Ingredient[] list) throws Exception
- c. int rateFlavor(Ingredient[] list) throws BadTasteException
- d. int rateFlavor(Ingredient[] list) throws BitterException

Majme nasledovnú definíciu triedy Widget:

```
1. class Widget extends Thingee {  
2.     static private int widgetCount = 0;  
3.     public String wName;  
4.     int wNumber;  
5.     static int addWidget() {  
6.         widgetCount++;  
7.         wName = "I am Widget # " + widgetCount;  
8.         return widgetCount;  
9.     }  
10.    public Widget() {  
11.        wNumber = addWidget();  
12.    }  
13. }
```

Čo sa stane po skompilovaní tejto triedy a použití viacerých inštancií Widget objektov v programe?

Select one or more:

- a. Nastane chyba pri kompliacii na riadku 11.
- b. Trieda sa skompiluje, každý objekt Widget bude mať jedinečnú hodnotu wNumber a wName v takom poradí v akom sa objekty vytvárajú.
- c. Program sa skompiluje, ale počas behu programu vznikne chyba súvisiaca s prístupom k premennej wName v metóde addWidget().
-  d. Nastane chyba pri kompliacii na riadku 7.

Aký bude výsledok po skompliovaní nasledovného kódu?

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List<Integer> glst1 = new ArrayList();  
4.         List nglst1 = glst1;  
5.         List nglst2 = nglst1;  
6.         List<Integer> glst2 = glst1;  
7.     }  
8. }
```

Select one or more:

- a. Kód sa skompliuje s varovnou správou na riadku 5.
- b. Kód sa skompliuje s varovnou správou na riadku 3.
- c. Kód sa skompliuje s varovnou správou na riadku 4.
- d. Kód sa skompliuje s varovnou správou na riadku 6.
- e. Kód sa skompliuje bez varovných správ.

Ktoré z deklarácií môžu byť použité na riadku 3, aby sa nasledovný program dal skompilovať a spustiť bez chyby?

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         // SEM VLOŽTE SPRÁVNU DEKLARÁCIU  
4.         for (int i = 0; i <= 5; i++) {  
5.             List<Integer> row = new ArrayList<Integer>();  
6.             for (int j = 0; j <= i; j++)  
7.                 row.add(i * j);  
8.             ds.add(row);  
9.         }  
10.        for (List<Integer> row : ds)  
11.            System.out.println(row);  
12.    }  
13. }
```

Select one or more:

- a. List<ArrayList<Integer>> ds = new ArrayList<ArrayList<Integer>>();
- b. List<List, Integer> ds = new List<List, Integer>();
- c. List<List<Integer>> ds = new List<List<Integer>>();
- d. List<List, Integer> ds = new ArrayList<List, Integer>();
- e. List<List<Integer>> ds = new ArrayList<List<Integer>>();
- f. ArrayList<ArrayList<Integer>> ds = new ArrayList<ArrayList<Integer>>();
- g. List<List, Integer> ds = new ArrayList<ArrayList, Integer>();
- h. List<List<Integer>> ds = new ArrayList<ArrayList<Integer>>();

Ktoré z uvedených príkazov sa nedajú použiť pre nasledovný kód?

```
1. class AClass<V> {  
2.     AClass() { System.out.println(this); }  
3.     <T> AClass(T t) { System.out.println(t); }  
4.     <T> AClass(T t, V v) { System.out.println(t + ", " + v); }  
5. }
```

Select one or more:

- a. AClass<String> ref5 = new <String>AClass<String>("one");
- b. AClass<String> ref4 = new <Integer>AClass<String>(2007);
- c. AClass<String> ref6 = new AClass<String>(2007, "one");
- d. AClass<String> ref8 = new <Integer>AClass<String>("one", 2007);
- e. AClass<String> ref7 = new <Integer>AClass<String>(2007, "one");
- f. AClass<String> ref3 = new AClass<String>(2007);
- g. AClass<String> ref2 = new AClass<String>("one");
- h. AClass<String> ref1 = new AClass<String>();

Nech `Thing` je trieda. Kolko objektov a kolko referencií je vytvorených nasledovným kódom?

1. `Thing item, stuff;`
2. `item = new Thing();`
3. `Thing entity = new Thing();`

Select one or more:

- a. Sú vytvorené tri referencie.
- b. Je vytvorený jeden objekt.
- c. Sú vytvorené dve referencie.
- d. Sú vytvorené tri objekty.
- e. Sú vytvorené dva objekty.
- f. Je vytvorená jedna referencia.

Čo vypíše nasledovný program po jeho spustení?

```
1. public class MyClass {  
2.     public static void main(String[] args) {  
3.         B b = new B("Test");  
4.     }  
5. }  
6. class A {  
7.     A() { this("1", "2"); }  
8.     A(String s, String t) { this(s + t); }  
9.     A(String s) { System.out.println(s); }  
10. }  
11. class B extends A {  
12.     B(String s) { System.out.println(s); }  
13.     B(String s, String t) { this(t + s + "3"); }  
14.     B() { super("4"); }  
15. }
```

Select one or more:

- a. Vypíše sa retázec "Test" nasledovaný retázcom "Test".
- b. Vypíše sa retázec "123" nasledovaný retázcom "Test".
- c. Vypíše sa retázec "12" nasledovaný retázcom "Test".
- d. Vypíše sa retázec "Test".
- e. Vypíše sa retázec "4" nasledovaný retázcom "Test".

Ktoré z deklarácií členskej premennej accounts je možné použiť na riadku 2 uvedeného kódu?

```
1. public class Testing {  
2.     // VLOŽTE SPRÁVNU DEKLARÁCIU ČLENSKEJ PREMENNEJ  
3.     public long getNum(String name) {  
4.         Long number = accounts.get(name);  
5.         return number == null ? 0 : number;  
6.     }  
7.     public void setNum(String name, long number) {  
8.         accounts.put(name, number);  
9.     }  
10. }
```

Select one or more:

- a. private Map<String, Long> accounts = new Map<String, Long>();
- b. private Map<String<Long>> accounts = new HashMap<String<Long>>();
- c. private Map accounts = new HashMap();
- d. private Map<String, long> accounts = new HashMap<String, long>();
- e. private Map<String, Long> accounts = new HashMap<String, Long>();

Čo sa stane ak sa pokúsime skompilovať a spustiť nasledovný kód? (Poznámka: Number je predok Integer.)

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List<? super Integer> sList = new ArrayList<Number>();  
4.         int i = 2007;  
5.         sList.add(i);  
6.         sList.add(++i);  
7.         Number num = sList.get(0);  
8.     }  
9. }
```

Select one or more:

- a. Kód sa skompiluje a spustí normálne.
- b. Kód sa skompiluje, ale po spustení program vyhodí výnimku ClassCastException na riadku 7.
- c. Kód sa neskompileuje kvôli chybe na riadku 3.
- d. Kód sa neskompileuje kvôli chybe na riadku 6.
- e. Kód sa neskompileuje kvôli chybe na riadku 7.

Question 1

Not yet
answeredMarked out of
3.00

Flag question

Aký je najmenší zoznam tried výnimiek, ktoré musí deklarovať klauzulou throws prekrytá metóda f() v nasledovnom kóde (riadok 10)?

```
1. class A {  
2.     // InterruptedException je priamy potomok Exception.  
3.     void f() throws ArithmeticException, InterruptedException {  
4.         div(5, 5);  
5.     }  
6.     int div(int i, int j) throws ArithmeticException {  
7.         return i / j;  
8.     }  
9.     public class MyClass extends A {  
10.        void f() /* throws DOPLŇTE ZOZNAM TRIED VÝNIMIEK */ {  
11.            try {  
12.                div(5, 0);  
13.            } catch (ArithmeticException e) {  
14.                return;  
15.            }  
16.            throw new RuntimeException("ArithmeticException was expected.");  
17.        }  
18.    }
```

Select one or more:

- a. Nemusí deklarovať žiadne výnimky.
- b. Musí deklarovať, že vyhadzuje RuntimeException.
- c. Musí deklarovať, že vyhadzuje ArithmeticException.
- d. Musí deklarovať, že vyhadzuje ArithmeticException a InterruptedException.
- e. Musí deklarovať, že vyhadzuje InterruptedException.

Question 4

Not yet
answered

Marked out of
3.00

Flag question

Ktoré z príkazov, keď ich vložíme na riadok 6, nespôsobia chybu kompliacie?

```
1. public class ThisUsage {  
2.     int planets;  
3.     static int suns;  
4.     public void gaze() {  
5.         int i;  
6.         // SEM VLOŽTE SPRÁVNY PRÍKAZ  
7.     }  
8. }
```

Select one or more:

- a. `i = suns;`
- b. `this = new ThisUsage();`
- c. `i = this.planets;`
- d. `this.i = 4;`
- e. `suns = planets;`

Majme vlastnú hierarchiu výnimiek odvodenú od `java.lang.Exception` nasledovne (`BitterException` a `SourException` sú potomkami `BadTasteException`):

`Exception`

```
+-- BadTasteException  
    +-- BitterException  
    +-- SourException
```

Tieto výnimky majú konštruktor s jediným parametrom typu reťazec (`String`). Majme metódu deklarovanú nasledovne:

```
int rateFlavor(Ingredient[] list) throws BadTasteException
```

Ktoré z nasledujúcich konštrukcií vyjadrujú správny spôsob vyhodenia jednej z týchto výnimiek v uvedenej metóde?

Select one or more:

- a. `throw new SourException("Ewww!");`
- b. `throw SourException("Ewww!");`
- c. `new SourException("Ewww!");`
- d. `throws new SourException("Ewww!");`

```
1. class Widget extends Thingee {  
2.     static private int widgetCount = 0;  
3.     public String wName;  
4.     int wNumber;  
5.     static int addWidget() {  
6.         widgetCount++;  
7.         wName = "I am Widget # " + widgetCount;  
8.         return widgetCount;  
9.     }  
10.    public Widget() {  
11.        wNumber = addWidget();  
12.    }  
13. }
```

Čo sa stane po skompilovaní tejto triedy a použití viacerých inštancií Widget objektov v programe?

Select one or more:

- a. Program sa skompiluje, ale počas behu programu vznikne chyba súvisiaca s prístupom k premennej wName v metóde addWidget().
- b. Nastane chyba pri kompliacii na riadku 7.
- c. Trieda sa skompiluje, každý objekt Widget bude mať jedinečnú hodnotu wNumber a wName v takom poradí v akom sa objekty vytvárajú.
- d. Nastane chyba pri kompliacii na riadku 11.

Question 8

Not yet
answered

Marked out of
3.00

Flag question

Ktoré z uvedených príkazov sa nedajú použiť pre nasledovný kód?

```
1. class AClass<V> {  
2.     AClass() { System.out.println(this); }  
3.     <T> AClass(T t) { System.out.println(t); }  
4.     <T> AClass(T t, V v) { System.out.println(t + ", " + v); }  
5. }
```

Select one or more:

- a. AClass<String> ref3 = new AClass<String>(2007);
- b. AClass<String> ref1 = new AClass<String>();
- c. AClass<String> ref8 = new <Integer>AClass<String>("one", 2007);
- d. AClass<String> ref5 = new <String>AClass<String>("one");
- e. AClass<String> ref2 = new AClass<String>("one");
- f. AClass<String> ref7 = new <Integer>AClass<String>(2007, "one");
- g. AClass<String> ref6 = new AClass<String>(2007, "one");
- h. AClass<String> ref4 = new <Integer>AClas<String>(2007);

Not yet
answered

Marked out of
3.00

Flag question

Sme vás vysledok po pokuse skompilovať a spustiť nasledovný kód?

```
1. public class Polymorphism {  
2.     public static void main(String[] args) {  
3.         A ref1 = new C();  
4.         B ref2 = (B) ref1;  
5.         System.out.println(ref2.g());  
6.     }  
7. }  
8. class A {  
9.     private int f() { return 0; }  
10.    public int g() { return 3; }           ↗  
11. }  
12. class B extends A {  
13.     private int f() { return 1; }  
14.     public int g() { return f(); }  
15. }  
16. class C extends B { public int f() { return 2; } }
```

Select one or more:

- a. Program sa skompiluje a po spustení vypíše 0.
- b. Program sa skompiluje a po spustení vypíše 3.
- c. Program sa skompiluje a po spustení vypíše 1.
- d. Program sa nepodarí skompilovať.
- e. Program sa skompiluje a po spustení vypíše 2.

- h. AClass<String> ref4 = new <Integer>AClas<String>(2007);

Majme triedy A, B a C, pričom B je odvodená od A a C je odvodená od B. Každá z týchto tried implementuje metódu doIt(). Ako môžem zavolať metódu doIt() z triedy A v objekte vytvorenom z triedy C?

Select one or more:



- a. this.super.doIt();
- b. A.this.doIt();
- c. ((A) this).doIt();
- d. super.doIt();
- e. doIt();
- f. super.super.doIt();
- g. Nie je to možné.

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

MyClass {

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

```
1. public class MyClass {  
2.     public static void main(String[] args) {  
3.         C c = new C();  
4.         System.out.println(c.max(13, 29));  
5.     }  
6. }  
7. class A {  
8.     int max(int x, int y) { if (x > y) return x; else return y; }  
9. }  
10. class B extends A{  
11.     int max(int x, int y) { return super.max(y, x) - 10; }  
12. }  
13. class C extends B {  
14.     int max(int x, int y) { return super.max(x + 10, y + 10); }  
15. }
```

Select one or more:

- a. Kód sa neskompiluje, pretože metóda `max()` v triede B posiela argumenty vo volaní `super.max(y, x)` v nesprávnom poradí.
- b. Kód sa skompiluje a po spustení vypíše 23.
- c. Kód sa skompiluje a po spustení vypíše 13.
- d. Kód sa neskompiluje, pretože volanie metódy `max()` nie je jednoznačné.
- e. Kód sa skompiluje a po spustení vypíše 39.
- f. Kód sa skompiluje a po spustení vypíše 29.

Ktoré z deklarácií členskej premennej accounts je možné použiť na riadku 2 uvedeného kódu?

```
1. public class Testing {  
2.     // VLOŽTE SPRÁVNU DEKLARÁCIU ČLENSKEJ PREMENNEJ  
3.     public long getNum(String name) {  
4.         Long number = accounts.get(name);  
5.         return number == null ? 0 : number;  
6.     }  
7.     public void setNum(String name, long number) {  
8.         accounts.put(name, number);  
9.     }  
10. }
```

Select one or more:

- a. private Map<String, Long> accounts = new Map<String, Long>();
- b. private Map<String<Long>> accounts = new HashMap<String<Long>>();
- c. private Map accounts = new HashMap();
- d. private Map<String, long> accounts = new HashMap<String, long>();
- e. private Map<String, Long> accounts = new HashMap<String, Long>();

Čo sa stane ak sa pokúsime skompilovať a spustiť nasledovný kód? (Poznámka: Number je predok Integer.)

```
1. public class Testing {  
2.     public static void main(String[] args) {  
3.         List<? super Integer> sList = new ArrayList<Number>();  
4.         int i = 2007;  
5.         sList.add(i);  
6.         sList.add(++i);  
7.         Number num = sList.get(0);  
8.     }  
9. }
```

Select one or more:

- a. Kód sa skompiluje a spustí normálne.
- b. Kód sa skompiluje, ale po spustení program vyhodí výnimku ClassCastException na riadku 7.
- c. Kód sa neskompileuje kvôli chybe na riadku 3.
- d. Kód sa neskompileuje kvôli chybe na riadku 6.
- e. Kód sa neskompileuje kvôli chybe na riadku 7.

Question 1

Not yet
answeredMarked out of
3.00

Flag question

Aký je najmenší zoznam tried výnimiek, ktoré musí deklarovať klauzulou throws prekrytá metóda f() v nasledovnom kóde (riadok 10)?

```
1. class A {  
2.     // InterruptedException je priamy potomok Exception.  
3.     void f() throws ArithmeticException, InterruptedException {  
4.         div(5, 5);  
5.     }  
6.     int div(int i, int j) throws ArithmeticException {  
7.         return i / j;  
8.     }  
9.     public class MyClass extends A {  
10.        void f() /* throws DOPLŇTE ZOZNAM TRIED VÝNIMIEK */ {  
11.            try {  
12.                div(5, 0);  
13.            } catch (ArithmeticException e) {  
14.                return;  
15.            }  
16.            throw new RuntimeException("ArithmeticException was expected.");  
17.        }  
18.    }
```

Select one or more:

- a. Nemusí deklarovať žiadne výnimky.
- b. Musí deklarovať, že vyhadzuje RuntimeException.
- c. Musí deklarovať, že vyhadzuje ArithmeticException.
- d. Musí deklarovať, že vyhadzuje ArithmeticException a InterruptedException.
- e. Musí deklarovať, že vyhadzuje InterruptedException.

Question 4

Not yet
answered

Marked out of
3.00

Flag question

Ktoré z príkazov, keď ich vložíme na riadok 6, nespôsobia chybu kompliacie?

```
1. public class ThisUsage {  
2.     int planets;  
3.     static int suns;  
4.     public void gaze() {  
5.         int i;  
6.         // SEM VLOŽTE SPRÁVNY PRÍKAZ  
7.     }  
8. }
```

Select one or more:

- a. `i = suns;`
- b. `this = new ThisUsage();`
- c. `i = this.planets;`
- d. `this.i = 4;`
- e. `suns = planets;`

Majme vlastnú hierarchiu výnimiek odvodenú od `java.lang.Exception` nasledovne (`BitterException` a `SourException` sú potomkami `BadTasteException`):

`Exception`

```
+-- BadTasteException  
    +-- BitterException  
    +-- SourException
```

Tieto výnimky majú konštruktor s jediným parametrom typu reťazec (`String`). Majme metódu deklarovanú nasledovne:

```
int rateFlavor(Ingredient[] list) throws BadTasteException
```

Ktoré z nasledujúcich konštrukcií vyjadrujú správny spôsob vyhodenia jednej z týchto výnimiek v uvedenej metóde?

Select one or more:

- a. `throw new SourException("Ewww!");`
- b. `throw SourException("Ewww!");`
- c. `new SourException("Ewww!");`
- d. `throws new SourException("Ewww!");`

```
1. class Widget extends Thingee {  
2.     static private int widgetCount = 0;  
3.     public String wName;  
4.     int wNumber;  
5.     static int addWidget() {  
6.         widgetCount++;  
7.         wName = "I am Widget # " + widgetCount;  
8.         return widgetCount;  
9.     }  
10.    public Widget() {  
11.        wNumber = addWidget();  
12.    }  
13. }
```

Čo sa stane po skompilovaní tejto triedy a použití viacerých inštancií Widget objektov v programe?

Select one or more:

- a. Program sa skompiluje, ale počas behu programu vznikne chyba súvisiaca s prístupom k premennej wName v metóde addWidget().
- b. Nastane chyba pri kompliacii na riadku 7.
- c. Trieda sa skompiluje, každý objekt Widget bude mať jedinečnú hodnotu wNumber a wName v takom poradí v akom sa objekty vytvárajú.
- d. Nastane chyba pri kompliacii na riadku 11.

Question 8

Not yet
answered

Marked out of
3.00

 Flag question

Ktoré z uvedených príkazov sa nedajú použiť pre nasledovný kód?

```
1. class AClass<V> {  
2.     AClass() { System.out.println(this); }  
3.     <T> AClass(T t) { System.out.println(t); }  
4.     <T> AClass(T t, V v) { System.out.println(t + ", " + v); }  
5. }
```

Select one or more:

- a. AClass<String> ref3 = new AClass<String>(2007);
- b. AClass<String> ref1 = new AClass<String>();
- c. AClass<String> ref8 = new <Integer>AClass<String>("one", 2007);
- d. AClass<String> ref5 = new <String>AClass<String>("one");
- e. AClass<String> ref2 = new AClass<String>("one");
- f. AClass<String> ref7 = new <Integer>AClass<String>(2007, "one");
- g. AClass<String> ref6 = new AClass<String>(2007, "one");
- h. AClass<String> ref4 = new <Integer>AClas<String>(2007);

Not yet
answered

Marked out of
3.00

Flag question

Sada vysledok po pokuse skompilovať a spustiť nasledovný kód?

```
1. public class Polymorphism {  
2.     public static void main(String[] args) {  
3.         A ref1 = new C();  
4.         B ref2 = (B) ref1;  
5.         System.out.println(ref2.g());  
6.     }  
7. }  
8. class A {  
9.     private int f() { return 0; }  
10.    public int g() { return 3; }           ↗  
11. }  
12. class B extends A {  
13.     private int f() { return 1; }  
14.     public int g() { return f(); }  
15. }  
16. class C extends B { public int f() { return 2; } }
```

Select one or more:

- a. Program sa skompiluje a po spustení vypíše 0.
- b. Program sa skompiluje a po spustení vypíše 3.
- c. Program sa skompiluje a po spustení vypíše 1.
- d. Program sa nepodarí skompilovať.
- e. Program sa skompiluje a po spustení vypíše 2.

- h. AClass<String> ref4 = new <Integer>AClas<String>(2007);

Majme triedy A, B a C, pričom B je odvodená od A a C je odvodená od B. Každá z týchto tried implementuje metódu doIt(). Ako môžem zavolať metódu doIt() z triedy A v objekte vytvorenom z triedy C?

Select one or more:



- a. this.super.doIt();
- b. A.this.doIt();
- c. ((A) this).doIt();
- d. super.doIt();
- e. doIt();
- f. super.super.doIt();
- g. Nie je to možné.

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

MyClass {

Aký bude výsledok po skompilovaní a spustení nasledovného programu?

```
1. public class MyClass {  
2.     public static void main(String[] args) {  
3.         C c = new C();  
4.         System.out.println(c.max(13, 29));  
5.     }  
6. }  
7. class A {  
8.     int max(int x, int y) { if (x > y) return x; else return y; }  
9. }  
10. class B extends A{  
11.     int max(int x, int y) { return super.max(y, x) - 10; }  
12. }  
13. class C extends B {  
14.     int max(int x, int y) { return super.max(x + 10, y + 10); }  
15. }
```

Select one or more:

- a. Kód sa neskompiluje, pretože metóda `max()` v triede B posiela argumenty vo volaní `super.max(y, x)` v nesprávnom poradí.
- b. Kód sa skompiluje a po spustení vypíše 23.
- c. Kód sa skompiluje a po spustení vypíše 13.
- d. Kód sa neskompiluje, pretože volanie metódy `max()` nie je jednoznačné.
- e. Kód sa skompiluje a po spustení vypíše 39.
- f. Kód sa skompiluje a po spustení vypíše 29.