


Email Address: djtesla@gmail.com
Date Started: 01/06/2021 4:59:57 PM
Date Completed: 01/06/2021 5:55:08 PM
Attempt: 1

Question Topic		
Num	Question	
	Respondent's Answer	Correct Answer
Java oo		
Group Earned Points: 17 of 25 (68%)		

1 Melyik állítás hamis az immutable osztályokkal kapcsolatban?

- ☐ Érdemes az attribútumait final kulcsszóval ellátni, hogy csak egyszer lehessen neki értéket adni.
- ☐ Használatuk azért is hasznos, mert paraméterként átadva biztosak lehetünk benne, hogy a hívás során nem változik az értéke.
- ☐ A String osztály immutable
- ☒ A metódusai módosíthatják az attribútumok értékét, csak a paraméterek értékét nem módosíthatják

- ☐ Érdemes az attribútumait final kulcsszóval ellátni, hogy csak egyszer lehessen neki értéket adni.
- ☐ Használatuk azért is hasznos, mert paraméterként átadva biztosak lehetünk benne, hogy a hívás során nem változik az értéke.
- ☐ A String osztály immutable
- ☒ A metódusai módosíthatják az attribútumok értékét, csak a paraméterek értékét nem módosíthatják

 Earned Points: 1 of 1

2

Adott az alábbi kódrészlet:

```
public class TrainerData {
    private String name;

    private List<String> courses;

    public TrainerData(String name, List<String> courses) {
        this.name = name;
        this.courses = new ArrayList<>(courses);
    }

    public String getName() {
        return name;
    }

    public List<String> getCourses() {
        return courses;
    }
}
```

Melyik hamis az alábbi állítások közül?

- ☐ () A TrainerData osztály name attribútuma osztályon kívülről módosítható, ugyanis nincs final módosítószóval ellátva
- ☐ () A TrainerData immutable, még a courses lista tartalma sem változtatható kívülről
- ☐ () A name attribútum módosítható, pl. így: new TrainerData("John Doe", Arrays.asList("Biology")).getName().toUpperCase();
- ☒ (X) Mindhárom hamis

- ☐ () A TrainerData osztály name attribútuma osztályon kívülről módosítható, ugyanis nincs final módosítószóval ellátva
- ☐ () A TrainerData immutable, még a courses lista tartalma sem változtatható kívülről
- ☐ () A name attribútum módosítható, pl. így: new TrainerData("John Doe", Arrays.asList("Biology")).getName().toUpperCase();
- ☒ (X) Mindhárom hamis



Earned Points: 1 of 1

3

Mely igaz a JavaBeans konvenciókkal kapcsolatban?

- ☐ () Minden Java osztálynak JavaBeansnek kell lennie
- ☐ () Kötelezően lenniük kell getter és setter metódusainak
- ☐ () Minden típus esetén ugyanazt az elnevezést kell alkalmaznunk, get prefix, utána az attribútum neve, aminek az első karaktere nagybetű, pl. getName()
- ☒ (X) Attribútum, és a hozzá tartozó metódusok összefoglaló neve property

- ☐ () Minden Java osztálynak JavaBeansnek kell lennie
- ☐ () Kötelezően lenniük kell getter és setter metódusainak
- ☐ () Minden típus esetén ugyanazt az elnevezést kell alkalmaznunk, get prefix, utána az attribútum neve, aminek az első karaktere nagybetű, pl. getName()
- ☒ (X) Attribútum, és a hozzá tartozó metódusok összefoglaló neve property



Earned Points: 1 of 1

4

Mit ír ki az alábbi kódrészlet?

```
public class WithName {

    private String name;

    public WithName(String name) {
        this.name = name;
    }

    public void modifyName(String name) {
        return this.name;
    }

    public static void main(String[] args) {
        System.out.println(new WithName("John Doe").modifyName("Jack Doe"));
    }
}
```

- ☒ (X) John Doe
- ☐ () Jack Doe
- ☐ () null
- ☐ () Nem fordul le

- ☐ () John Doe
- ☐ () Jack Doe
- ☐ () null
- ☒ (X) Nem fordul le



Earned Points: 0 of 1

5

Mit ír ki az alábbi kódrészlet?

```
public class Numbers {

    public String addFive(String s) {
        return s + 5;
    }

    public static void main(String[] args) {
        System.out.println(new Numbers().addFive(10));
    }
}
```

- ☒ (X) 105
- ☐ () 15
- ☐ () Nem fordul le
- ☐ () Futás közben kivételt dob

- ☐ () 105
- ☐ () 15
- ☒ (X) Nem fordul le
- ☐ () Futás közben kivételt dob



Earned Points: 0 of 1

6

Mit ír ki az alábbi kódrészlet?

```
public class NameConverter {
```

```
    public void convertNames(String name, List<String> otherNames) {
        name = name.toUpperCase();
        for (int i = 0; i < otherNames.size(); i++) {
            otherNames.set(i, otherNames.get(i).toUpperCase());
        }
    }
}
```

```
    public static void main(String[] args) {
        String name = "John Doe";
        List<String> names = new ArrayList<>(Arrays.asList("Jane Doe"));
        new NameConverter().convertNames(name, names);
        System.out.print(name);
        System.out.print(" ");
        System.out.print(names);
    }
}
```

- ☐ () John Doe [JANE DOE]
- ☒ (X) JOHN DOE [JANE DOE]
- ☐ () John Doe [Jane Doe]
- ☐ () [John Doe, Jane Doe]

- ☒ (X) John Doe [JANE DOE]
- ☐ () JOHN DOE [JANE DOE]
- ☐ () John Doe [Jane Doe]
- ☐ () [John Doe, Jane Doe]



Earned Points: 0 of 1

7

Mit ír ki az alábbi kódrészlet?

```
public class NameConverter {
```

```
    public String convertName(String name) {
        if (name.equals("")) {
            return;
        }
        return name.toUpperCase();
    }
}
```

```
    public static void main(String[] args) {
        System.out.println(new NameConverter().convertName(""));
    }
}
```

- ☒ (X) Nem fordul le
- ☐ () Üres string, azaz ""
- ☐ () null
- ☐ () Futás közben hiba keletkezik

- ☒ (X) Nem fordul le
- ☐ () Üres string, azaz ""
- ☐ () null
- ☐ () Futás közben hiba keletkezik



Earned Points: 1 of 1

8

Adott a következő kódrészlet:

```
public List<Course> createCourses(String... names) {
    List<Course> courses = new ArrayList<>();
    // Bejárás
}
```

Melyik a helytelen módja a paraméterek bejárásának?

- ☒ () for (String name: names) { courses.add(new Course(name)); }
- ☐ () for (int i = 0; i
- ☒ (X) for (int i = 0; i
- ☐ () Egyik megadási mód sem helyes

- ☐ () for (String name: names) { courses.add(new Course(name)); }
- ☐ () for (int i = 0; i
- ☒ (X) for (int i = 0; i
- ☐ () Egyik megadási mód sem helyes



Earned Points: 1 of 1

9

Mit ír ki az alábbi kódrészlet?

```
public class Builder {

    private String name;

    public Builder setName(String name) {
        this.name = name;
        return this;
    }

    public String build() {
        return name;
    }

    public static void main(String[] args) {
        System.out.print(new Builder().build());
        System.out.print(" ");
        System.out.print(new Builder().setName("John Doe").setName("Jack Doe").build());
    }
}
```

- ☒ () Nem fordul le
- ☒ (X) null Jack Doe
- ☐ () Jack Doe
- ☐ () null John Doe

- ☐ () Nem fordul le
- ☒ (X) null Jack Doe
- ☐ () Jack Doe
- ☐ () null John Doe



Earned Points: 1 of 1

10

Melyik metódussal lehet túlterhelni (overload) a következő metódust?

```
public Course createCourse(String name) {}
```

- ☐ public Course create(String name) {}
- ☐ public SpecificCourse createCourse(String name) {}
- ☒ public Course createCourse(String name, Level level) {}
- ☐ public Course createCourse(String name) {}

- ☐ public Course create(String name) {}
- ☐ public SpecificCourse createCourse(String name) {}
- ☒ public Course createCourse(String name, Level level) {}
- ☐ public Course createCourse(String name) {}



Earned Points: 1 of 1

11

Adott az alábbi kódrészlet.

```
public class Concatenator {
```

```
    public static String convert(int a, int b) {
        return Integer.toString(a) + Integer.toString(b);
    }
```

```
    public static void main(String[] args) {
        // ???
    }
```

```
}
```

A main metódusból hogyan lehet meghívni a convert metódust?

- ☐ Csak a Concatenator.convert(5, 6); utasítással
- ☐ Csak a convert(5, 6); utasítással
- ☐ Csak a new Concatenator().convert(5, 6); utasítással
- ☒ Mindhárom utasítással, de nem mindegyik javasolt

- ☐ Csak a Concatenator.convert(5, 6); utasítással
- ☐ Csak a convert(5, 6); utasítással
- ☐ Csak a new Concatenator().convert(5, 6); utasítással
- ☒ Mindhárom utasítással, de nem mindegyik javasolt



Earned Points: 1 of 1

12

Mit ír ki az alábbi kódrészlet?

```
public class State {

    private static int instance = 0;

    public State() {
        instance++;
    }

    public static void main(String[] args) {
        new State();
        new State();
        System.out.println(instance);
    }

}
```

- ☐ () Nem fordul le
- ☐ () 0
- ☐ () 1
- ☒ (X) 2

- ☐ () Nem fordul le
- ☐ () 0
- ☐ () 1
- ☒ (X) 2



Earned Points: 1 of 1

13

Mit ír ki az alábbi kódrészlet? Feltételezzük, hogy a két osztály két külön .java fájlban van deklarálva, a megfelelő könyvtárakban.

```
package foo;
public class Foo {
    public static int value = 10;
}
package bar;

import static foo.Foo;

public class Bar {
    public static void main(String[] args) {
        System.out.println(value);
    }
}
```

- ☐ () Nem fordul le
- ☒ (X) 10
- ☐ () 0
- ☐ () Futás közben hiba

- ☒ (X) Nem fordul le
- ☐ () 10
- ☐ () 0
- ☐ () Futás közben hiba



Earned Points: 0 of 1

14

Adott a következő kódrészlet:

```
public class Trainer {

    private String name;

    public Trainer(String name) {
        this.name = name;
    }
}
```

Mely állítás hamis az alábbiak közül?

- ☒ (X) Az osztály példányosítható a következő módon:
Trainer t = new Trainer();
- ☐ () Az osztály példányosítható a következő módon:
Trainer t = new Trainer("Anonymous");
- ☐ () Az osztály példányosítható a következő módon:
Trainer t = new Trainer(null);
- ☐ () Nem kerül legenerálásra paraméter nélküli implicit konstruktor, hiszen van explicit konstruktor

- ☒ (X) Az osztály példányosítható a következő módon:
Trainer t = new Trainer();
- ☐ () Az osztály példányosítható a következő módon:
Trainer t = new Trainer("Anonymous");
- ☐ () Az osztály példányosítható a következő módon:
Trainer t = new Trainer(null);
- ☐ () Nem kerül legenerálásra paraméter nélküli implicit konstruktor, hiszen van explicit konstruktor



Earned Points: 1 of 1

15

public class Trainer {

```
    private String name;
```

```
    private int age;
```

```
    public Trainer(String name) {
        this(name, 0);
    }
}
```

```
    public Trainer(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

Az alábbi konstruktorok melyik nem illeszhető be a fenti osztályba harmadik konstruktorként?

- ☐ () public Trainer() {this(null, 0);}
- ☐ () public Trainer() {super();}
- ☒ (X) public Trainer() {String name = "anonymous"; this(name);}
- ☐ () public Trainer(int age) { this(null, age); }

- ☐ () public Trainer() {this(null, 0);}
- ☐ () public Trainer() {super();}
- ☒ (X) public Trainer() {String name = "anonymous"; this(name);}
- ☐ () public Trainer(int age) { this(null, age); }



Earned Points: 1 of 1

16

Mit ír ki az alábbi kódrészlet?

```
class A {
    String s = "-";
    protected A() { this("d"); s += "a"; }
    private A(String e) { s += "d"; }
}

class B extends A {
    B() { super(); s += "b"; }
}

class C extends B {
    C() { s += "c"; }
    public static void main(String... boo) {
        System.out.println((new C()).s);
    }
}
```

- ☐ -dabc
- ☐ -abc
- ☐ -adbc
- ☒ -cba

- ☐ (X) -dabc
- ☐ () -abc
- ☐ () -adbc
- ☐ () -cba



Earned Points: 0 of 1

17

Mely állítás igaz az implicit (default) konstruktorral kapcsolatban?

- ☐ () Fogadhat paramétereket
- ☐ () private módosítószóval rendelkezik
- ☐ () Amennyiben definiálunk egy paraméteres konstruktort, az implicit (default) konstruktor továbbra is hívható
- ☒ (X) Törzsében szerepel egy super() hívás

- ☐ () Fogadhat paramétereket
- ☐ () private módosítószóval rendelkezik
- ☐ () Amennyiben definiálunk egy paraméteres konstruktort, az implicit (default) konstruktor továbbra is hívható
- ☒ (X) Törzsében szerepel egy super() hívás



Earned Points: 1 of 1

18

Mit ír ki az alábbi kódrészlet?

```
class Ex1 {
    public static void main(String[] args) {
        int a[] = { 1,2,3,4}; // 1
        System.out.print(a instanceof Object); // 2
    }
}
```

- ☒ (X) true
- ☐ () false
- ☐ () Nem fordul le az egyessel jelölt sor
- ☐ () Nem fordul le a kettessel jelölt sor

- ☒ (X) true
- ☐ () false
- ☐ () Nem fordul le az egyessel jelölt sor
- ☐ () Nem fordul le a kettessel jelölt sor



Earned Points: 1 of 1

19

Adott a következő kódrészlet:

```
public class Human {

    private String name;

}
```

Ha létrehozunk egy leszármazottat (Trainer extends Human), melyik állítás hamis?

- ☐ Leszármazottban elérhető a name attribútum, ha a Human osztályban protected láthatósági módosítóval látjuk el
- ☐ Definiáljunk egy publikus gettert az attribútumnak, és akkor a leszármazottban is hozzá lehet férni a getterrel
- ☒ A super.name hívással is hozzá lehet férni a leszármazottban
- ☐ A protected módosítószó a package private kiterjesztése, használatakor nem csak az azonos csomagban lévő osztályok, hanem bármely leszármazott eléri az adott tagot

- ☐ Leszármazottban elérhető a name attribútum, ha a Human osztályban protected láthatósági módosítóval látjuk el
- ☐ Definiáljunk egy publikus gettert az attribútumnak, és akkor a leszármazottban is hozzá lehet férni a getterrel
- ☒ A super.name hívással is hozzá lehet férni a leszármazottban
- ☐ A protected módosítószó a package private kiterjesztése, használatakor nem csak azonos csomagban lévő osztályok, hanem bármely leszármazott eléri az adott tagot



Earned Points: 1 of 1

20

Adott a következő kódrészlet:

```
class Mid {
    public int findMid(int n1, int n2) {
        return (n1 + n2) / 2;
    }
}

public class Calc extends Mid {
    public static void main(String[] args) {
        int n1 = 22, n2 = 2;
        // Egészítsd ki
        System.out.println(n3);
    }
}
```

Mit kell a megjegyzés helyére írni, hogy 12 értéket írja ki?

- ☒ Calc c = new Calc(); int n3 = c.findMid(n1,n2);
- ☐ int n3 = super.findMid(n1,n2);
- ☐ Calc c = new Mid(); int n3 = c.findMid(n1, n2);
- ☐ int n3 = this.findMid(n1,n2);

- ☒ Calc c = new Calc(); int n3 = c.findMid(n1,n2);
- ☐ int n3 = super.findMid(n1,n2);
- ☐ Calc c = new Mid(); int n3 = c.findMid(n1, n2);
- ☐ int n3 = this.findMid(n1,n2);



Earned Points: 1 of 1

21

Adott az alábbi kódrészlet:

```
class Plant {
    String getName() { return "plant"; }
    Plant getType() { return this; }
}
```

```
class Flower extends Plant {
    // Hiányzó kód
}
```

```
class Tulip extends Plant { }
```

Melyik sor nem illeszthető a hiányzó kód helyére?

- ☒ (X) Tulip getType() { return new Tulip(); }
- ☐ () Plant getType() { return this; }
- ☐ () Object getType() { return this; }
- ☐ () Flower getType() { return this; }

- ☐ () Tulip getType() { return new Tulip(); }
- ☐ () Plant getType() { return this; }
- ☒ (X) Object getType() { return this; }
- ☐ () Flower getType() { return this; }



Earned Points: 0 of 1

22

Mely deklaráció fordul le?

- ☒ (X) public abstract class Canine { public void speak(); }
- ☐ () public class Canine abstract { public abstract void speak(); }
- ☐ () public class Canine { public abstract void speak(); }
- ☐ () public abstract class Canine { public void speak() { } }

- ☐ () public abstract class Canine { public void speak(); }
- ☐ () public class Canine abstract { public abstract void speak(); }
- ☐ () public class Canine { public abstract void speak(); }
- ☒ (X) public abstract class Canine { public void speak() { } }



Earned Points: 0 of 1

23

Adottak a következő állítások. Az A és E osztály. A B és D interfész. A C absztrakt osztály.
Melyik helytelen a következő megadások közül?

- ☐ () class F implements B { }
- ☒ (X) class F extends A, E { }
- ☐ () class F extends E { }
- ☐ () class F implements B,D { }

- ☐ () class F implements B { }
- ☒ (X) class F extends A, E { }
- ☐ () class F extends E { }
- ☐ () class F implements B,D { }



Earned Points: 1 of 1

24

Adottak a következő kódrészletek.

```
abstract class X {
    public abstract void methodX();
}
```

```
interface Y {
```

```
}
```

Melyik kódrészlet helytelen?

- ☐ class Z extends X implements Y { public void methodZ(); }
- ☒ abstract class Z extends X implements Y { public void methodZ() {} }
- ☐ class Z extends X implements Y { public void methodX() {} }
- ☐ abstract class Z extends X implements Y { }

- ☒ class Z extends X implements Y { public void methodZ(); }
- ☐ abstract class Z extends X implements Y { public void methodZ() {} }
- ☐ class Z extends X implements Y { public void methodX() {} }
- ☐ abstract class Z extends X implements Y { }



Earned Points: 0 of 1

25

Melyik helyes deklaráció interfészen belül?

- ☒ public void doMore(long bow);
- ☐ public void doMore(long bow) {}
- ☐ private short hop = 23;
- ☐ public Name();

- ☒ public void doMore(long bow);
- ☐ public void doMore(long bow) {}
- ☐ private short hop = 23;
- ☐ public Name();



Earned Points: 1 of 1

Respondent Summary

Overall Time Used: 00:55:11
 Score Percentile: 36th
 Overall Result: Pass

Final Score: 68%