Java 2 pótvizsga

Respondent Complete Analysis

Assessment Unique ID: A0GRQK10SZVS

Email Address:	djtesla@gmail.com
Név:	Zámbó Ernő
Date Started:	01/11/2021 6:26:07 PM
Date Completed:	01/11/2021 7:24:19 PM
Attempt:	1

Question Topic		
Num	Question	
	Respondent's Answer	Correct Answer

Java oo

Group Earned Points: 25 of 25 (100%)

1

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

2

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

3

```
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 illeszthető 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

4

Adott az alábbi két osztály kódja.

```
class BaseClass {
   private float x = 1.0F;
   void setF(float y) {this.x = 2 * y;}
}

class ChildClass extends BaseClass {
   private float x = 2.0F;
   // overriding method goes here
}
```

Az alábbi kódrészletek közül melyik illeszthető a komment helyére, hogy érvényes felülírást (overriding) hozzunk létre?

( ) public void setF(double y) {this.x = (float) 2 * y;}
(X) public void setF(float y) {this.x = 2 * y;}
( ) private void setF(float y) {this.x = 2 * y;}
( ) float setF(float y) {this.x = 2 * y; return x;}

( ) public void setF(double y) {this.x = (float) 2 * y;}
(X) public void setF(float y) {this.x = 2 * y;}
( ) private void setF(float y) {this.x = 2 * y;}
( ) float setF(float y) {this.x = 2 * y; return x;}

Earned Points: 1 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));
  }

}
```

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

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

Earned Points: 1 of 1

6 Adott az alábbi két osztály:

```
class A {
   private int i;
   public A(int x){
      this.i = x;
   }
}

class B extends A {
    private int j;
    public B(int x, int j){
        super(x);
        this.j = j;
    }
}
```

A B osztályban az alábbiak közül mely konstruktort lehet még elhelyezni anélkül, hogy fordítási hiba keletkezzen?

( ) public B(){}
( ) public B(int j) {this.j = j;}
( ) public B(int j) {super.i = j; this.j = j;}
(X) code>public B(int j) {super(j + 1); this.j = j;}

( ) public B(){}
( ) public B(int j) {this.j = j;}
( ) public B(int j) {super.i = j; this.j = j;}
(X) code>public B(int j) {super(j + 1); this.j = j;}

Earned Points: 1 of 1

7 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?

( ) Tulip getType() { return new Tulip(); }
( ) Plant getType() { return this; }
(X) 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: 1 of 1

8 Melyik helyes deklaráció interfészen belül?

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

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

Earned Points: 1 of 1

9 Jelöld meg a helyes állítást!
Az egységbezárás biztosítja,

( ) hogy az osztályok funkciókat örökölhessenek más osztályoktól.
(X) hogy az osztályok csak kiválasztott attribútumokat és metódusokat tegyenek hozzáférhetővé más osztályok számára.
( ) hogy az osztályok adott metódusokat absztraktként deklaráljanak és azokat az alosztályok implementálhassák szükség szerint.
( ) hogy az osztály egy metódusa, amely X osztály példányát fogadja paraméterként, az X osztály leszármazottját is fogadhassa paraméterként.

( ) hogy az osztályok funkciókat örökölhessenek más osztályoktól.
(X) hogy az osztályok csak kiválasztott attribútumokat és metódusokat tegyenek hozzáférhetővé más osztályok számára.
( ) hogy az osztályok adott metódusokat absztraktként deklaráljanak és azokat az alosztályok implementálhassák szükség szerint.
( ) hogy az osztály egy metódusa, amely X osztály példányát fogadja paraméterként, az X osztály leszármazottját is fogadhassa paraméterként.

Earned Points: 1 of 1

10 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

11 Adott a következő osztály. Hogyan érhetjük el, hogy az id elérhető legyen, de ne legyen módosítható se osztályon kívülről, se belülről?

```
public class Immutable{
    public int id;
    public Immutable(int id){this.id = id;}
}
```

( ) Írjuk át private láthatóságra!
( ) Legyen az id static és final, és legyen egy public static int getId() metódus, amely visszaadja az értékét!
(X) Legyen az id private és final, és legyen egy public int getId() metódus, amely visszaadja az értékét!
( ) Legyen az id protected láthatóságú!

( ) Írjuk át private láthatóságra!
( ) Legyen az id static és final, és legyen egy public static int getId() metódus, amely visszaadja az értékét!
(X) Legyen az id private és final, és legyen egy public int getId() metódus, amely visszaadja az értékét!
( ) Legyen az id protected láthatóságú!

Earned Points: 1 of 1

12 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

13 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"));
  }

}
```

( ) John Doe
( ) Jack Doe
( ) null
(X) Nem fordul le

( ) John Doe
( ) Jack Doe
( ) null
(X) Nem fordul le

Earned Points: 1 of 1

14 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
(X) 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
(X) Mindhárom utasítással, de nem mindegyik javasolt

Earned Points: 1 of 1

15 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

16 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
(X) 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
(X) A metódusai módosíthatják az attribútumok értékét, csak a paraméterek értékét nem módosíthatják

Earned Points: 1 of 1

17 Adott a következő kód:

```
public class ValueChange {
   private int value = 0;
   public void stepOne(int value) {
      value = value;
   }
   public void stepTwo(int value) {
      this.value = value;
   }
   public static void main(String[] args) {
      ValueChange vc = new ValueChange();
      vc.stepOne(100);
      System.out.print(vc.value);
      vc.stepTwo(200);
      System.out.println(vc.value);
   }
}
```

A program futásakor mi kerül kiírásra?

( ) 0 100
( ) 100 100
(X) 0 200
( ) 100 200

( ) 0 100
( ) 100 100
(X) 0 200
( ) 100 200

Earned Points: 1 of 1

18 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?

(X) 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);

(X) 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

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
(X) 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
(X) 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

Earned Points: 1 of 1

20 Adott az alábbi két osztály. A példányosítás az osztályoktól függetlenül, a megadott kódrészlet szerint történik. Mit ír ki a program futtatáskor?

```
public class Programmer extends Employee {
    private double s;
    private double b;

    public Programmer(double s, double b) {
        super(s);
        this.s = s;
        this.b = b;
    }

    public double getSalary() {
        return s + b;
    }
}

class Employee {
    private double s;
    public Employee(double salary) {
        this.s = s;
    }
    public double getSalary() {
        return s;
    }
}
...
        Programmer p = new Programmer(200000, 30000);
        Employee e = new Programmer(300000, 50000);
        System.out.print(p.getSalary());
        System.out.print(" ");
        System.out.print(e.getSalary());
...
```

( ) 230000.0 300000.0
( ) 230000 350000
(X) 230000.0 350000.0
( ) ClassCastException keletkezik futásidőben.

( ) 230000.0 300000.0
( ) 230000 350000
(X) 230000.0 350000.0
( ) ClassCastException keletkezik futásidőben.

Earned Points: 1 of 1

21 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);
    }
}
```

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

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

Earned Points: 1 of 1

22 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);
  }
}
```

(X) Nem fordul le
( ) 10
( ) 0
( ) Futás közben hiba

(X) Nem fordul le
( ) 10
( ) 0
( ) Futás közben hiba

Earned Points: 1 of 1

23 Adott az alábbi kódrészlet és abban a következő metódusok:
void printSum(int a, int b){ System.out.println("int sum: "+(a+b)); }
void printSum(Integer a, Integer b){ System.out.println("Integer sum: "+(a+b)); }
void printSum(double a, double b){ System.out.println("double sum: "+(a+b));}
A következő metódus hívása esetén mi kerül kiírásra?
printSum(1, 2);

(X) int sum: 3
( ) Integer sum: 3
( ) double sum: 3.0
( ) double sum: 3

(X) int sum: 3
( ) Integer sum: 3
( ) double sum: 3.0
( ) double sum: 3

Earned Points: 1 of 1

24 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

25 Melyik metódussal lehet túlterhelni (overload) a következő metódust?
public Course createCourse(String name) {}

( ) public Course create(String name) {}
( ) public SpecificCourse createCourse(String name) {}
(X) public Course createCourse(String name, Level level) {}
( ) public Course createCourse(String name) {}

( ) public Course create(String name) {}
( ) public SpecificCourse createCourse(String name) {}
(X) public Course createCourse(String name, Level level) {}
( ) public Course createCourse(String name) {}

Earned Points: 1 of 1

Respondent Summary

Overall Time Used:	00:58:12
Score Percentile:	100th
Overall Result:	Pass

Final Score: **100%**