
A "java Villa -v" parancs jelentése:

- A java interpreter elindítja a Villa osztály statikus main metódusát, és átadja neki paraméterként a "-v" stringet.
- A java interpreter elindítja először a Villa osztály statikus main metódusát, majd megpróbálja elindítani a -v nevű osztály statikus main metódusát, de a -v nem egy szabályos osztálynév, ezért a parancs hibaüzenettel megszakad.
- A java interpreter a "-v", azaz verbose opcióval indul, és végrehajtja a Villa osztály statikus main metódusát.

Helyes-e az alábbi kódrészlet?

```
int i = 1;
i = i * 3 + 1;
int j;
j = i + 1;
```

- Nem.
- Igen.

Hányféleképpen lehet Javaban megjegyzést írni?

- 1
- 4
- 3
- 2

Helyes-e az alábbi kódrészlet?

```
int i = 1, s = 0;
while (i < 10) {
  s += i;
  i++;
}
```

- Nem.
- Igen.

Helyes-e az alábbi kódrészlet?

```
float f = 3;
double d = (double)f;
```

- Igen.
- Nem.

Melyek helyesek?

```
interface X {
  public int i = 1;
}
```

```
public interface Y {
   int alma() { return 1; }
}
```

- Mindkettő.
- Y
- Egyik sem.
- X

Mennyivel egyenlő a `(new A()).alma(z)` hívás visszatérési értéke?

```
class X {}
interface Y {}
class Z extends X implements Y {}
class A {
  int alma( X x ){ return 1; }
  int alma( Y y ){ return 2; }
}
...
Z z = new Z();
```

- 2-vel.
- 1-gyel.
- A program helytelen.

Helyes-e az alábbi program?

```
abstract class A {
  abstract int alma();
  static void cseresznye(){
    System.out.println((new A).alma());
  }
}
```

- Igen.
- Nem.

Helyes-e az alábbi program?

```
class A {
  int alma(){ return 1; }
  int alma( int i ){ return i; }
}
```

- Nem.
- Igen.

Helyes-e az alábbi program? Ha igen, mit ír a print? Ha nem, miért nem?

```
class A { int i = 2; }
class B extends A {
  int i = 4;
  void print(){ System.out.print(i); }
}
```

- Helytelen, mert öröklődés során nem engedett a példányváltozók felüldefiniálása.
- Helyes, 4-et ír ki.
- Helyes, 2-t ír ki.
- Helytelen, mert nem egyértelmű a print számára, hogy melyik i-re történt hivatkozás.

Az interfészek adattagjai mindig statikusak?

- Soha.
- Csak esetenként.
- Igen.

Mire nyújt lehetőséget az öröklődés mechanizmusa Java-ban?

- Többszörös altípusképzése, és kódöröklésre.
- Többszörös altípusképzése, és ha nincs konfliktus, többszörös kódöröklésre.
- Csak egyszeres altípusképzésre és kódöröklésre.
- Többszörös altípusképzése, de csak egyszeres kódöröklésre.

Egy konstruktor deklarációjában a specifikált visszatérési típus...

- ... akármi lehet.
- ... void.
- ... helyére nem kerül semmi.
- ... a konstruktort tartalmazó osztály.

Létezik-e olyan osztály, ami minden más osztálynak az őse.

- Nem.
- Igen.

Melyik helyes az alábbi három osztálydefiníció közül?

```
class A {
  int x = 12;
  int y = -x;
}
```

```
class A {
  int x;
  { x = 12; }
  int y;
  { y = -x; }
}
```

```
class A {
  int x;
  { x = 12; }
  int y = -x;
}
```

- Egyik sem.
- Csak az első.
- Mindegyik.
- Csak az első és a második.

Mit jelent a *static* módosítószó egy változódeklarációban?

- A változó osztályszintű.
- A változót csak statikusan, a saját osztályból lehet meghivatkozni.
- A változó statikus, nem módosítható értékű.
- A változót egyszerre csak egy végrehajtási szál használhatja.

Helyes-e az alábbi program?

```
class A {
  int alma( char c ){ return c; }
  int alma( byte b ){ return b; }
}
```

- Nem.
- Igen.

Mennyit ad vissza az a.alma() metódushívás?

```
final class A { int alma(){ return 1; } }
final class B extends A { int alma(){ return 2; } }
...
A a = new B();
```

- 2-t.
- 1-et.
- A program helytelen.

Egy változó dinamikus típusa a statikus típusának egy leszármazottja, vagy maga a statikus típus.

- Nem.
- Igen.

Ha az A osztály a B osztály leszármazottja, akkor egy A típusú változó hivatkozhat tetszőleges B osztályú objektumra.

- Igen.
- Nem.

Melyek helyesek az alábbi deklarációk közül?

```
int[] x = {1,2,3,4};
int y[] = new int[3];
```

- Az első.
- Egyik sem.
- Mindkettő.
- A második.

Legyen adott az alábbi fordítási egység.

```
package a.b;
public class A {}
```

Melyek jók az alábbiak közül, ha a fenti A osztályból szeretnénk leszármaztatni?

```
import a.*;
class B extends A {}
```

```
import a.*;
class B extends b.A {}
```

- Egyik sem.
- Mindkettő.
- A baloldali.
- A jobbldali.

Legyen a két forrásfájlunk tartalma az alábbi. Lefordulnak-e?

```
package a;
class A { int x = 1; }

package a.b;
import a.*;
class B extends A { int y = x; }
```

- Igen.
- Nem.

Helyes-e az alábbi két interfész-definíció?

```
interface A { void alma() throws java.io.IOException; }
interface B extends A { void alma() throws Exception; }
```

- Igen.
- Nem.

Helyes-e az alábbi metódusdefiníció?

```
void alma() throws Exception { throw new java.io.IOException(); }
```

- Nem.
- Igen.

Helyes-e az alábbi metódusdefiníció?

```
void alma( int x ) throws java.io.IOException {
  if( x==0 ){ throw new java.io.IOException(); }
  else{ throw new Exception(); }
}
```

- Nem.
- Igen.

Lehet-e hivatkozni valahogyan az 1 értékű x változóra a kiír() metóduson belül?

```
class A {
  int x = 1;
  class B {
    int x = 2;
    void kiír( int x ){ ... }
  }
}
```

- A fenti kódrészlet eleve hibás.
- Igen.
- A kettős elfedés miatt nem.

Tegyük fel, hogy adott az alábbi interfész-definíció.

```
interface I { int x(); }
```

Egy A osztályon belül definiáljuk az alábbi metódust.

```
I alma( final int x ){
  class X implements I { public int x() { return x; } };
  return new X();
}
```

Mit gondolunk a (new A()).alma(7).x() kifejezésről?

- Hibás, mert az x változó az alma() metódus lokális változója, amelynek hatásköre az alma() metódusra korlátozódik, így a metódusból való kilépés után nem hivatkozható.
- Értelmes, de definiálatlan értékű. (Implementáció-függő az értéke.)
- Hibás, mert az x hívásának pillanatában nincs értelmes értéke az x változónak.
- Értelmes, 7-tel egyenlő.

Két logikai típusú kifejezés diszjunkcióját...

- az "or" operátorral képezzük.
- a && operátorral képezzük.
- a + operátorral képezzük.
- a || operátorral képezzük.

Helyes-e az alábbi kódrészlet?

```
short s = 3;
byte b = (byte)s;
```

- Nem.
- Igen.

Mi lesz az x értéke?

```
boolean b=false;
int x = 3;
if (b=true) x++;
else x = 8;
```

- 4
- 8
- 3

Ha az A osztály a B osztály leszármazottja, akkor az alábbi helyes:

```
B b = new A();
```

- Nem.
- Igen.

```
Object o = new Integer(42);
```

- Ez hibás.
- Ez helyes.

Inicializátor blokkot csak példányváltozó inicializálására lehet használni.

- Igaz.
- Hamis.

Mennyivel egyenlő a (new A()).alma(o) hívás visszatérési értéke?

```
class A {
  int alma( Object o ){ return 1; }
  int alma( String s ){ return 2; }
}
...
Object o = new String();
```

- 2-vel.
- 1-gyel.

Helyes-e az alábbi program?

```
abstract class A {
  abstract int alma();
  static void cseresznye(){
    System.out.println((new A).alma());
  }
}
```

- Igen.
- Nem.

Ha az A osztály a B osztály leszármazottja, akkor az alábbi helyes:

```
A a = new B();
```

- Igen.
- Nem.

Ha egy attribútum definiálásakor nem adunk meg láthatósági módosítószót, akkor az az attribútum...

- csak ugyanazon objektumon belül hivatkozható.
- csak ugyanazon osztályon belül hivatkozható.
- csak az azonos csomagban definiált osztályokon belül hivatkozható.
- csak ugyanazon osztályon és a leszármazott osztályokon belül hivatkozható.

Helyes-e az alábbi program? Ha igen, mit ír a print? Ha nem, miért nem?

```
class A { int i = 2; }
class B extends A {
  int i = 4;
  void print(){ System.out.print(i); }
}
```

- Helyes, 4-et ír ki.
- Helytelen, mert nem egyértelmű a print számára, hogy melyik i-re történt hivatkozás.
- Helyes, 2-t ír ki.
- Helytelen, mert öröklődés során nem engedett a példányváltozók felüldefiniálása.

Java-ban lehet írni statikus inicializátor blokkot. (Azaz statikus változó inicializálására.)

- Igaz.
- Hamis.

Mennyit ad vissza az a.alma() metódushívás?

```
class A { protected int alma() { return 1; } }
class B extends A { public int alma() { return 2; } }
...
A a = new B();
```

- 2-t.
- A program helytelen.
- 1-et.

Mi lesz az *egyenlőek* függvény visszatérési értéke?

```
class A {
  static int alma() { return 1; }
  static int cseresznye() { return alma(); }
}
class B extends A {
  static int alma() { return 2; }
  static boolean egyenlőek(){
    A a = new B();
    return a.cseresznye() == cseresznye();
  }
}
```

- false
- true
- A program helytelen.

Melyek igazak az alábbiak közül? (a) A kétdimenziós tömböket Java-ban egydimenziós tömbök tömbjeként készíthetjük el. (b) Egy kétdimenziós tömb minden sora egyenlő hosszúságú Java-ban.

- Egyik sem
- Mindkettő
- (b)
- (a)

Legyen a két forrásfájlunk tartalma az alábbi. Lefordulnak-e?

```
package a;
class A { private int x = 1; }

package a.b;
import a.A;
class B extends A { int y = x; }
```

- Igen.
- Nem.

Hány fordítási egység kerülhet egy forrásfájlba?

- Több
- Egy

Melyik esetben lesz több x értéke?

```
try {
  x = 1;
  throw new Exception();
} catch( Exception e ){
  x = 2;
} finally {
  x = 3;
}
```

```
try {
  x = 1;

} catch( Exception e ){
  x = 2;
} finally {
  x = 3;
}
```

- Egyenlő lesz, mindkét esetben 3.
- A baloldali kódrészlet hibás.
- A jobboldali esetben.
- A baloldali esetben.

Mennyi lesz x értéke az alábbi kód végrehajtása után?

```
try {
  x = 1;
  throw new java.io.IOException();
} catch( Exception e ){
  x += 2;
} catch( Throwable e ){
  x += 3;
}
```

- 4
- 3
- 1
- A kód rossz.

Helyes-e az alábbi metódusdefiníció?

```
void alma( int x ) throws java.io.IOException {
  if( x==0 ){ throw new java.io.IOException(); }
  else{ throw new RuntimeException(); }
}
```

- Nem.
- Igen.

Helyes-e az alábbi programrészlet?

```
void alma() {
  Object o = new T(){};
}
```

- Igen, de csak akkor, ha T egy nem absztrakt osztály.
- Igen, helyes lehet, legyen a T egy osztály vagy akár egy interfész.
- Igen, de csak akkor, ha a T egy osztály.
- Nem.

Melyik a leginkább igaz az alábbi állítások közül? Ha az A osztály két példánymetódusa szinkronizált, akkor

- a két metódus A osztályú vagy annak leszármazott osztályába tartozó objektumokra meghívva nem hajtódhat végre konkurrensen.
- a két metódust ugyanarra az objektumra meghívva nem hajtódhatnak végre konkurrensen.
- a két metódust A osztályú objektumokra meghívva nem hajtódhatnak végre konkurrensen.

ID 2 3 Melyek igazak az alábbi állítások közül? 1. A Thread osztály yield() művelete elavult. 2. A Thread osztály sleep() művelete InterruptedException kivételt válthat ki.

- Az első.
- Mindkettő.
- A második.
- Egyik sem.

Az alábbiak közül mely lehet az "Alakzat" nevű publikus interfész forráskódját tartalmazó fájl neve?

- alakzat.java
- Alakzat.class
- alakzat.class
- Alakzat.java

Mi lesz az x értéke?

```
int x = 3;
if (x==3)
  x++;
  if (x==6) x--;
else
  x = 2;
```

- 3
- 2
- 5
- 4

Helyes-e az alábbi kódrészlet?

```
int i = 1, s = 0;
while (i < 10) {
  s += i;
  i++;
}
```

- Nem.
- Igen.

Helyes-e az alábbi kódrészlet?

```
byte b = 3;
short s = b;
```

- Igen.
- Nem.

Helyes-e az alábbi kódrészlet?

```
int i = 1;
i = i * 3 + 1;
int j;
j = i + 1;
```

- Nem.
- Igen.

A típusok hierarchiája egy fát alkot.

- Nem igaz.
- Igaz.

Ha az A osztály a B osztály leszármazottja, akkor az alábbi helyes:

```
A a = new B();
```

- Igen.
- Nem.

Ha az A osztály a B osztály leszármazottja, akkor az alábbi helyes:

```
B b = new A();
```

- Nem.
- Igen.

Helyes-e az alábbi program?

```
class A {
  int alma(){ return 1; }
  int alma( int i ){ return i; }
}
```

- Nem.
- Igen.

Ha az A osztály a B osztály leszármazottja, akkor egy B típusú változó hivatkozhat tetszőleges A osztályú objektumra.

- Nem.
- Igen.

Helyes-e az alábbi program?

```
abstract class A {
  abstract int alma();
  void cseresznye(){
    System.out.println(alma());
  }
}
```

- Nem.
- Igen.

Helyes-e az alábbi program?

```
class A {
  int alma( Object o ){ return 1; }
  int alma( String s ){ return 2; }
}
```

- Igen.
- Nem.

Java-ban nincs többszörös kódöröklődés.

- Nem igaz.
- Igaz.

```
class A implements X {}
class B implements Y {}
class C extends A {}
class D implements Z {}
class E extends D implements V {}
interface X {}
interface Y {}
interface Z extends Y, V {}
interface V {}
interface W extends V {}
```

- Az X altípusa a D-nek.
- Az E altípusa az Y-nak.
- A C altípusa az Y-nak.
- Az E altípusa a W-nek.

```
A a = new B();
```

Itt A az "a" dinamikus típusa, és B a statikus típusa.

- Nem.
- Igen.

Az interfészek adattagjai mindig módosíthatatlanok?

- Csak esetenként.
- Igen.
- Soha.

Melyek helyesek az alábbiak közül?

```
Object[] t = new Integer[3];            Integer[] t = new Object[3];
```

- A jobboldali.
- A baloldali.
- Egyik sem.
- Mindkettő.

Mit kell írni a * helyére?**

```
import java.io.IOException;
interface A { void alma() throws IOException; }
interface B { void alma() throws Exception; }
interface C extends A, B {}
class D implements C { public void alma() *** {} }
```

- throws Exception
- throws Throwable
- throws IOException

Helyes-e az alábbi metódusdefiníció?

```
void alma() throws java.io.IOException { throw new Exception(); }
```

- Igen.
- Nem.

Mennyi lesz x értéke az alábbi kód végrehajtása után?

```
try {
  x = 1;
  throw new Throwable ();
} catch( Exception e ){
  x += 2;
} finally {
  x += 3;
}
```

- 1
- 4
- 3
- A kód rossz.

Melyek lehetnek helyesek az alábbi kifejezések közül?

```
this.x          x.this          this.this          x.x
```

- Csak az első.
- A harmadik kivételével mindegyik.
- Mindegyik.
- A két szélső.

Melyik igaz az alábbi két állítás közül? 1. T lehet egy interfész. 2. Az o dinamikus típusa T, ha T egy osztály.

```
void alma() {
  Object o = new T(){};
}
```

- Egyik sem.
- A második.
- Mindkettő.
- Az első.

Mi a probléma az alábbi osztálydefinícióval?

```
class A {
  Object lock = new Object();
  void alma(){
    try { lock.wait(); }
    catch (Exception e) {}
  }
}
```

- Az alma() specifikációjában fel kell tüntetni további olyan kivételeket, amelyeket a wait() metódus kiválthat, de még nincsenek lekezelve.
- A wait() metódus meghívásakor a végrehajtási szál nem rendelkezik a lock objektum zárjával.
- A wait() metódusnak nem adtuk meg, hogy milyen feltételre kell várnia a programnak a lock objektumnál.
- A notify() műveletet nem hívja meg senki, ezért a program a végtelenségig fog várakozni.

Mivel hozunk létre egy végrehajtási szálat?

- Konstruktor meghívásával.
- Osztály betöltésével.
- run() metódus meghívásával.
- start() metódus meghívásával.

Mi lehet annak a fájlnak a neve, amelybe a "Teve" nevű publikus osztály forráskódját írjuk?

- teve.java
- Teve.java
- mindegy
- Teve.class

Helyes-e az alábbi kódrészlet?

```
double d = 3;
float f = d;
```

- Nem.
- Igen.

Helyes-e az alábbi kódrészlet?

```
double u = 1.0;
u = u * 3.2 + 1.78;
double v;
v = u + 1.3;
```

- Nem.
- Igen.

Mennyivel egyenlő *a.cseresznye()* visszatérési értéke?

```
class A {
  int alma() { return 1; }
  int cseresznye() { return alma(); }
}
class B extends A {
  int alma() { return 2; }
}
...
A a = new B();
```

- A program helytelen.
- 1
- Az a.cseresznye() kifejezés kiértékelése futási hibát okoz.
- 2

Mennyit ad vissza az *a.alma()* metódushívás?

```
final class A { int alma(){ return 1; } }
final class B extends A { int alma(){ return 2; } }
...
A a = new B();
```

- 1-et.
- 2-t.
- A program helytelen.

Melyek helyesek?

```
public interface X {
  static int i = 1;
}
```

- X
- Egyik sem.

```
interface Y {
   final int j = 2;
}
```

- Y
- Mindkettő.

Mi az x az alábbi példában?

```
String x;
```

- Egy referencia változó, amely String típusú objektumra mutathat.
- Egy String osztályú objektum, vagy annak leszármazottja.

Az interfészek <u>metódusai</u> mindig statikusak?

- Soha.
- Csak esetenként.
- Igen.

```
class A implements X {}
class B implements Y {}
class C extends A {}
class D implements Z {}
class E extends D implements V {}
interface X {}
interface Y {}
interface Z extends Y, V {}
interface V {}
interface W extends V {}
```

- Az E altípusa az Y-nak.
- A C altípusa az Y-nak.
- Az X altípusa a D-nek.
- Az E altípusa a W-nek.

Java-ban lehet írni példányszintű inicializátor blokkot. (Azaz példányváltozó inicializálására.)

- Hamis.
- Igaz.

Mennyit ad vissza az a.alma() metódushívás?

```
class A { int alma(){ return 1; } }
class B extends A { short alma(){ return 2; } }
...
A a = new B();
```

- A program helytelen.
- 1-et.
- 2-t.

Az interfészek adattagjai mindig módosíthatatlanok?

- Soha.
- Igen.
- Csak esetenként.

A String az Object egy altípusa.

- Igen.
- Nem.

Legyen a két forrásfájlunk tartalma az alábbi. Lefordulnak-e?

```
package a;
public class A { protected int x = 1; }

package a.b;
import a.A;
class B extends A { int y = x; }
```

- Nem.
- Igen.

Jól használja-e a throws kulcsszót az alábbi kódrészlet?

```
try { throws new Exception(); }
catch( Exception e ){}
```

- Igen.
- Nem.

Mennyit ír ki a kiír metódus?

```
class A {
  int x = 1;
  static class B { int x = 2; }
  static void kiír(){
    A.B b = new A.B();
    System.out.println(b.x);
  }
}
```

- Nem helyes a program, mert a hivatkozás x-re nem megfelelő.
- 1-et.
- Nem helyes a program, mert b létrehozása nem megfelelő.
- 2-t.

Tegyük fel, hogy adott az alábbi interfész-definíció.

```
interface I { int x(); }
```

Egy A osztályon belül definiáljuk az alábbi metódust.

```
I alma( final int x ){
  class X implements I { public int x() { return x; } };
  return new X();
}
```

Mit gondolunk a (new A()).alma(7).x() kifejezésről?

- Értelmes, de definiálatlan értékű. (Implementáció-függő az ertéke.)
- Hibás, mert az x változó az alma() metódus lokális változója, amelynek hatásköre az alma() metódusra korlátozódik, így a metódusból való kilépés után nem hivatkozható.
- Értelmes, 7-tel egyenlő.
- Hibás, mert az x hívásának pillanatában nincs értelmes értéke az x változónak.

Melyek igazak az alábbi állítások közül? 1. A Java futtató környezet garantálja, hogy a végrehajtási szálak nem kerülnek holtpontba az erőforrások megosztott használata során. 2. A Java futtató környezet biztosítja, hogy a végrehajtási szálak ne használhassák ugyanazt az erőforrást egyidejűleg.

- Az első.
- A második.
- Mindkettő.
- Egyik sem.

Mi a probléma az alábbi osztálydefinícióval?

```
class A {
  Object lock = new Object();
  void alma(){
    try { lock.wait(); }
    catch (Exception e) {}
  }
}
```

- A wait() metódus meghívásakor a végrehajtási szál nem rendelkezik a lock objektum zárjával.
- A notify() műveletet nem hívja meg senki, ezért a program a végtelenségig fog várakozni.
- Az alma() specifikációjában fel kell tüntetni további olyan kivételeket, amelyeket a wait() metódus kiválthat, de még nincsenek lekezelve.
- A wait() metódusnak nem adtuk meg, hogy milyen feltételre kell várnia a programnak a lock objektumnál.
