Verificare 02

Petculescu Mihai-Silviu

Capitol 03 - Test grilă

Răspunsuri

```
 1 - A ;  2 - A ;  3 - C ;  4 - A ;  5 - C ;
 6 - A ;  7 - A ;  8 - D ;  9 - A ; 10 - E ;
11 - B ; 12 - A ; 13 - A ; 14 - E ; 15 - C ;
16 - E ; 17 - E ; 18 - E ; 19 - A ; 20 - C ;
21 - C ; 22 - A ; 23 - A ; 24 - C ; 25 - A ;
26 - D ; 27 - B ; 28 - A ; 29 - A ; 30 - C ;
31 - B ; 32 - C ; 33 - C ;
```

Întrebarea 3.7.1.

Fie următorul program Java:

```
class C1 {
  int x = 1;
  C1() {
    System.out.print("x =" + x);
  }
}
class C2 extends C1 {
  int y = 3;
  C2(int y) {
    this.y = y;
  }
}
public class Test {
  public static void main(String[] args) {
    C2 obiect = new C2();
    System.out.println(" y = " + obiect.y);
  }
}
```

Ce puteţi spune despre acesta?
a) Eroare la compilare: nu există constructor fără argumente.
b) Programul este corect şi la execuţie va afişa: y = 3.
c) Programul este corect şi la execuţie va afişa: x = 1 y = 3.

> Răspuns: A

Întrebarea 3.7.2.

Fie următorul program Java:

```
class C0 {
  int x = 1;
}
class C1 {
  int y = 2;
}
class C2 extends C0, C1 {
  int z = 3;
}
public class TestDoi {
  public static void main(String[] args) {
    C2 object = new C2();
    System.out.println(object.x + object.y + object.z);
```

```
  }
}
```

Ce puteţi spune despre acesta?
a) Eroare la compilare: clasa C2 nu poate fi derivată direct din două clase.
b) Eroare la compilare: clasa C2 nu are constructor fără argumente.
c) Programuleste corect şi la execuţie va afişa: 6.

> Răspuns: A

Întrebarea 3.7.3.

Fie următorul program Java:

```
class x {
  private int x = 1;
  void x() {
    System.out.print(x + " ");
  }
}
class y extends x {
  private int x = 2;
  void x() {
    super.x();
    System.out.print(x);
  }
}
public class TestAtribute {
  public static void main(String[] args) {
    y obiect = new y();
    obiect.x();
  }
}
```

Ce puteţi spune despre acesta?
a) Eroare la compilare: nu se pot defini un atribut şi o metodă cu acelaşi nume (x).
b) Eroare la compilare: nu se poate defini un atribut cu numele clasei (x).
c) Eroare la compilare: nu se poate defini un atribut cu acelaşi nume într-o clasă derivată (x).
d) Programul este corect şi la execuţie va afişa: 1 2.

> Răspuns: C

Întrebarea 3.7.4.

Ce se va afişa la execuţia următorului program?

```
public class TestStatic {
  public static void main(String[] args) {
    Exemplu a = new Exemplu();
    Exemplu b = new Exemplu();
    System.out.print("a.x = " + a.x);
    a.x = 100; b.x = 200;
    System.out.print("a.x = " + a.x);
  }
}
class Exemplu {
  static int x = 0;
  Exemplu() { x++; };
}
```

a) a.x = 2 a.x = 200
b) a.x = 0 a.x = 100
c) a.x = 1 a.x = 100
d) a.x = 1 a.x = 101
e) a.x = 2 a.x = 100

> Răspuns: A

Întrebarea 3.7.5.

Ce puteţi spune despre următorul program?

```
public class Test {
  int i = 3, j = 4;
  public static void main(String[] args) {
    System.out.println(i ++ + ++ j)
  }
}
```

a) Va apărea eroare la compilare, deoarece parametrul metodei `main()` trebuie să fie `String args[]`.
b) Va apărea eroare la compilare, deoarece nu s-au inserat paranteze între operatorii ++, adică (i++) şi (++j).
c) Va apărea eroare la compilare, deoarece din funcţia statică `main()` nu putem accesa variabilele nestatice `i` şi `j`.
d) La execuţie se va afişa 8.
e) La execuţie se va afişa 9.

> Răspuns: C

Întrebarea 3.7.6.

Ce puteţi afirma despre următorul program?

```
interface I1 { float x = 2.3f; }
public class Test implements I1 {
  public static void main(String[] args) {
    System.out.print(x + " ");
    x = 6.7f;
    System.out.println(x);
  }
}
```

a) Va apărea eroare la compilare, deoarece valoarea variabilei x nu se mai poate modifica.
b) La execuţie se va afişa 2.3f 6.7f.
c) La execuţie se va afişa 2.3f 2.3f.
d) La execuţie se va afişa 2.3 6.7.
e) La execuţie se va afişa 2.3 2.3.

> Răspuns: A

Întrebarea 3.7.7.

Ce puteţi spune despre următorul program Java?

```
interface I1 { int x = 2; }
interface I2 extends I1 { int y = 3; }
class C1 { int y = 4; }
public class C3 extends C1 implements I2 {
  public static void main(String[] args) {
    System.out.println("x = " + x + ", y = " + y)
  }
}
```

a) Va apărea eroare la compilare, deoarece atributul y este ambiguu pentru C1 şi I2;
b) La execuţie se va afişa x = 2, y = 3.
c) La execuţie se va afişa x = 2, y = 4.

> Răspuns: A

Întrebarea 3.7.8.

Ce puteţi afirma despre următorul program Java?

```
class C1 {
  public String f() {
    return this.getClass().getName();
  }
}
abstract class C2 extends C1 {
  int x = 2;
  public abstract String f();
}
class C3 extends C2 {
  int y = 3;
  public String f() {
    return super.f() + ", x = " + x + ", y = " + y;
  }
}
public class TestAbstract {
```

```
  public static void main(String args[]) {
    C3 obiect = new C3();
    System.out.println(obiect.f());
  }
}
```

a) Programul este corect şi va afişa la execuţie: C3, x = 2, y = 3,
b) Programul este corect şi va afişa la execuţie: C1, x = 2, y= 3,
c) Va apărea eroare la compilare, deoarece în clasa `C2` s-a suprascris metoda `f()` ne-abstractă cu o metodă abstractă.
d) Va apărea eroare la compilare, deoarece apelul `super.f()` din clasa `C3` se referă la o metodă abstractă.
e) Va apărea eroare la execuţie, deoarece se intră într-o recursie infinită în metoda `f()` din clasa `C3`.

> Răspuns: D

Întrebarea 3.7.9.

Ce modificare trebuie făcută următorului program Java pentru a fi corect?

```
abstract class C1 {
  int x = 2;
  public final abstract String f();
}
final class C2 extends C1 {
  int y = 3;
  public String f() {
    return "x = " + x + ", y = " + y;
  }
}
public class TestFinal {
  public static void main(String args[]) {
    C2 obiect = new C2();
    System.out.println(obiect.f());
  }
}
```

a) Trebuie şters cuvântul `final` din definiţia metodei `f()` a clasei `C1`.
b) Trebuie şters cuvântul `final` din definiţia clasei `C2`.
c) Trebuie şters cuvântul `abstract` din definiţia clasei `C1`.
d) Trebuie ştearsă iniţializarea atributului `x` din declaraţia `int x = 2`; a clasei `C1`.
e) Programul este corect şi la execuţie va afişa x = 2, y = 3.

> Răspuns: A

Întrebarea 3.7.10.

Ce puteţi spune despre următorul program Java?

```
class C1 {
  int x = 1;
  void f(int x) { this.x = x; }
  int getX_C1() { return x; }
}
```

```
class C2 extends C1 {
  float x = 5.0f;
  int f(int x) { super.f((int) x); }
  float getX_C2() { return x; }
}
public class SuprascriereSiAscundere {
  public static void main(String[] args) {
    C2 obiect = new C2();
    obiect.f(4);
    System.out.print(obiect.getX_C2() + " ");
    System.out.println(obiect.getX_C1());
  }
}
```

a) Programul este corect şi va afişa la execuţie: 5.0 4.
b) Programul este corect şi va afişa la execuţie: 5 4.
c) Programul este corect şi va afişa la execuţie: 4.0 4.
d) Va apărea eroare la compilare, deoarece în clasa `C2` s-a suprascris greşit atributul `x` din clasa `C1`.
e) Va apărea eroare la compilare, deoarece metoda suprascrisă `f()` din clasa `C2` întoarce un tip diferit de `void`.

> Răspuns: E

Întrebarea 3.7.11.

Ce puteţi afirma despre următorul program Java?

```
class C1 {
  static String f() { return "Mesajul Unu din C1"; }
  String g() { return "Mesajul Doi din C1"; }
}
class C2 extends C1 {
  static String f() { return "Mesajul Unu din C2"; }
  String g() { return "Mesajul Doi din C2"; }
}
public class Test {
  public static void main(String[] args) {
    C1 obiect = new C2();
    System.out.println(obiect.f() + ", " + obiect.g());
  }
}
```

a) Programul este corect şi va afişa la execuţie: Mesajul Unu din C1, Mesajul Doi din C1.
b) Programul este corect şi va afişala execuţie: Mesajul Unu din C1, Mesajul Doi din C2.
c) Programul este corect şi va afişa la execuţie: Mesajul Unu din C2, Mesajul Doi din C1.
d) Programul este corect şi va afişa la execuţie: Mesajul Unu din C2, Mesajul Doi din C2.
e) Va apărea eroare la compilare, deoarece în clasa `Test` variabila obiect nu aparţine clasei `C2`.

> Răspuns: B

Întrebarea 3.7.12.

Ce puteţi spune despre următorul program Java?

```
class C1 {
  double x = 5.0f;
  void f(int x) {
    f((double) x + 2);
  }
  void f(double x) {
    this.x = x;
  }
}
public class Supraincarcare {
  public static void main(String args[]) {
    C1 obiect = new C1();
    obiect.f(4.0);
    System.out.println(obiect.x);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 4.0.
b) Programul este corect şi va afişa la execuţie: 5.0.
c) Programul este corect şi va afişa la execuţie: 6.0.
d) Va apărea eroare la compilare, deoarece există ambiguitate în alegerea metodei `f()`.
e) Va apărea eroare la execuţie, deoarece metoda `f(int)` intră într-o recursie infinită.

> Răspuns: A

Întrebarea 3.7.13.

Cercetaţi corectitudinea următorului program Java:

```
class C1 {
  static int f() { return j; }
  static int i = f();
  static int j = 1;
}
public class TestStatic {
  public static void main(String[] args) {
    System.out.println(C1.i);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 0.
b) Programul este corect şi va afişa la execuţie: 1.
c) Va apărea eroare la compilare, deoarece nu există nici un obiect declarat al clasei `C1`.
d) Va apărea eroare la compilare, deoarece membrii clasei `C1` nu sunt într-o ordine corectă.
e) Va apărea eroare la compilare, deoarece s-a omis declararea clasei `C1` ca fiind static.

> Răspuns: A

Întrebarea 3.7.14.

Ce puteţi afirma despre următorul program Java (liniile sunt numerotate)?

```
1. public class SubiectLicentaDoi {
2.   static int x = 10;
3.   static { x += 5; }
4.   public static void main (String args[]) {
5.     System.out.println("x =" + x);
6.   }
7.   static { x /= 5; }
8. }
```

a) Liniile 3 şi 7 nu se vor compila, deoarece lipsesc numele metodelor şi tipurile returnate.
b) Linia 7 nu se va compila, deoarece poate exista doar un iniţializator `static`.
c) Codul se compilează şi execuţia produce ieşirea x = 10.
d) Codul se compilează şi execuţia produce ieşirea x = 15.
e) Codul se compilează şi execuţia produce ieşirea x = 3.

> Răspuns: E

Întrebarea 3.7.15.

Ce se va afişa la execuţia următorului program Java?

```
public class SuprascriereSupraincarcare extends Baza {
  public void functie(int j) { System.out.print(" j = " + j); }
  public void functie(String s) { System.out.print(" s = " + s); }
  public static void main(String args[]) {
    Baza b1 = new Baza();
    Baza b2 = new SuprascriereSupraincarcare();
    b1.functie(5);
    b2.functie(6);
  }
}
class Baza {
  public void functie(int i) { System.out.print(" i = " + i); }
}
```

a) i = 5 i = 6.
b) j = 5 j = 6.
c) i = 5 j = 6.
d) s = 5 s = 6.
e) Eroare la compilare, deoarece definiţia clasei `Baza` apare după definiţia clasei `SupracriereSupraincarcare`.
f) Eroare la compilare, deoarece lipseşte cuvântul virtual din metoda `functie()` a clasei `Baza`.

> Răspuns: C

Întrebarea 3.7.16.

Ce puteţi spune despre următorul program Java?

```
class C1 {
  static int x = 1;
  int y = 2;
  static C1() { x = 4; }
  C1(int y) { this.y = y + 1; }
}
public class Test {
  public static void main(String args[]) {
    System.out.print(C1.x + " ");
    C1 obiect = new C1(5);
    System.out.println(obiect.y);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 1 6.
b) Programul este corect şi va afişa la execuţie: 4 6.
c) Programul este corect şi va afişa la execuţie: 1 2.
d) Programul este corect şi va afişa la execuţie: 4 2.
e) Va apărea eroare la compilare, deoarece s-a declarat unul din constructorii clasei `C1` ca fiind `static`.

> Răspuns: E

Întrebarea 3.7.17.

Ce puteţi afirma despre următorul program Java?

```
class C1 {
  int x = 1;
  C1(int x) { this.x = x; }
}
class C2 extends C1 {
  int y = 3;
  C2(int x) { this(x, y); }
  C2(int x, int y) { super(x); this.y = y; }
  public String toString() { return "x = " + x + ", y = " + y; }
}
public class Test {
  public static void main(String args[]) {
    C2 obiectUnu = new C2(4);
    C2 obiectDoi = new C2(5, 6);
    System.out.print(obiectUnu + " ");
    System.out.println(obiectDoi);
  }
}
```

a) Programul este corect şi va afişa la execuţie: x = 4, y = 3 x = 5, y = 6.
b) Programul este corect şi va afişa la execuţie: x = 1, y = 3 x = 5, y = 6.
c) Programul este corect şi va afişa la execuţie: x = 1, y = 3 x = 1, y = 6.
d) Programul este corect şi va afişa la execuţie: x = 1, y = 3 x = 1, y = 3.

e) Va apărea eroare la compilare, deoarece primul constructor al clasei `C2` conţine un apel explicit în care al doilea parametru este o variabilă instanţă.

> Răspuns: E

Întrebarea 3.7.18.

Indicaţi dacă programul următor este corect şi ceea ce se va afişa la execuţia acestuia:

```
class C1 {
  int x = 1;
  C1(int x) { this.x = x; }
}
class C2 extends C1 {
  int y = 2;
  public String toString() { return "x = " + x + ", y = " + y; }
}
public class Test {
  public static void main(String args[]) {
    C2 obiectUnu = new C2();
    C2 obiectDoi = new C2(3);
    System.out.print(obiectUnu);
    System.out.print(obiectDoi);
  }
}
```

a) Programul este corect şi va afişa la execuţie: x = 1, y = 2 x = 1, y = 2.
b) Programul este corect şi va afişa la execuţie: x = 1, y = 2 x = 3, y = 2.
c) Programul este corect şi va afişa la execuţie: x = 1, y = 3 x = 1, y = 3.
d) Programul este corect şi va afişa la execuţie: x = 1, y = 3 x = 3, y = 2.
e) Va apărea eroare la compilare, deoarece clasa `C1` nu are constructor fără parametri.

> Răspuns: E

Întrebarea 3.7.19.

Stabiliţi dacă programul următor este corect sau nu:

```
interface I1 {
  int j = 3;
  int[] i = { j + 1, j + 2 };
}
interface I2 {
  int j = 5;
  int[] i = { j + 2, j + 4 };
}
public class Test implements I1, I2 {
  public static void main(String[] args) {
    System.out.print(I1.j + " ");
    System.out.print(I2.i[1] + " ");
    System.out.print(I1.i[0]);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 3 9 4.
b) Va apărea eroare la compilare, deoarece interfeţele `I1` şi `I2` conţin declaraţiile aceloraşi atribute.
c) Va apărea eroare la compilare, deoarece în interfeţele `I1` şi `I2` atributul `i` conţine referinţă la atributul `j`.
d) Va apărea eroare la compilare, deoarece interfeţele `I1` şi `I2` conţin declaraţiile unui atribut de tip tablou.
e) Va apărea eroare la compilare, deoarece atributele interfeţelor `I1` şi `I2` nu se pot accesa dintr-o metodă statică.

> Răspuns: A

Întrebarea 3.7.20.

Ce puteţi spune despre corectitudinea următorului program Java?

```
interface I0 { int x; }
interface I1 extends I0 {
  int[] y = { x + 1, x + 2 };
}
interface I2 extends I0 {
  int[] y = { x + 3, x + 4 };
}
interface I3 extends I1, I2 {
  int z = I1.y[0], t = I2.y[1];
}
public class TestAtribute implements I3 {
  public static void main(String[] args) {
    I0.x = 3;
    System.out.print(x + " ");
    System.out.print(z + " ");
    System.out.print(t);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 3 4 7.
b) Va apărea eroare la compilare, deoarece interfeţele `I1` şi `I2` conţin declaraşiile aceloraşi atribute.
c) Va apărea eroare la compilare, deoarece în interfaţa `I0` atributul `x` nu este iniţializat.
d) Va apărea eroare la compilare, deoarece interfaţa `I3` accesează un atribut ambiguu.
e) Va apărea eroare la compilare, deoarece atributele interfeţelor `I0` şi `I3` nu se pot accesa dintr-o metodă statică.

> Răspuns: C

Întrebarea 3.7.21.

Ce puteţi afirma despre următorul program Java?

```
class ExceptiaNoastraUnu extends Exception {
  ExceptiaNoastraUnu() { super(); }
}
class ExceptiaNoastraDoi extends Exception {
  ExceptiaNoastraDoi() { super(); }
```

```
}
interface I1 {
  int f(int x) throws ExceptiaNoastraUnu, ExceptiaNoastraDoi;
}
interface I2 extends I1 {
  int f(int x) throws ExceptiaNoastraDoi;
}
class C1 implements I1 {
  public int f(int x) throws ExceptiaNoastraUnu,
    ExceptiaNoastraDoi {
      if (x < 0) throw new ExceptiaNoastraUnu();
      return x;
    }
  public int f(int x) throws ExceptiaNoastraDoi {
    if (x < 0) throw new ExceptiaNoastraDoi();
    return x;
  }
}
public class TestSuprascriereMetode {
  public static void main(String[] args) {
    C1 obiect = new C1();
    try {
      System.out.print(obiect.f(2) + " ");
      System.out.print(obiect.f(-2));
    } catch (ExceptiaNoastraUnu e) {
      System.out.print("A aparut o exceptie: " + e);
    } catch (ExceptiaNoastraDoi e) {
      System.out.print("A aparut o exceptie: " + e);
    }
  }
}
```

a) Programul este corect şi va afişa la execuţie: 2 A aparut o exceptie: ExceptiaNoastraUnu.
b) Programul este corect şi va afişa la execuţie: 2 A aparut o exceptie: ExceptiaNoastraDoi.
c) Va apărea eroare la compilare, deoarece metoda `f()` din interfaţa `I2` intră în contradicţie cu metoda `f()` din `I1`.
d) Va apărea eroare la compilare, deoarece interfețele `I1` şi `I2` nu conţin atributul `x`.
e) Va apărea eroare la compilare, deoarece în clasa `C1` nu se poate implementa metoda `f()` din interfaţa `I1`.

> Răspuns: C

Întrebarea 3.7.22.

Selectaţi varianta corectă referitoare la corectitudinea programului Java de mai jos:

```
interface I1 { int f(int x); }
interface I2 extends I1 { float f(float x); double f(double x); }
class C1 implements I2 {
  public int f(int x) { return x; }
  public float f(float x) { return x; }
  public double f(double x) { return x; }
}
public class TestSupraincarcareMetode {
```

```
  public static void main(String[] args) {
    C1 obiect = new C1();
    System.out.print(obiect.f(2) + " ");
    System.out.print(obiect.f(4.5) + " ");
    System.out.print(obiect.f(2.5f));
  }
}
```

a) Programul este corect şi va afişa la execuţie: 2 4.5 2.5.
b) Programul este corect şi va afişa la execuţie: 2 4.5 2.5f.
c) Va apărea eroare la compilare, deoarece metodele `f()` din interfaţa `I2` intră în contradicţie cu metoda `f()` din `I1`.
d) Va apărea eroare la compilare, deoarece metodele `f()` din interfaţa `I2` intră în contradicţie între ele.
e) Va apărea eroare la compilare, deoarece metodele `f()` din interfeţele `I1` şi `I2` nu sunt declarate public.

> Răspuns: A

Întrebarea 3.7.23.

Cercetaţi corectitudinea următorului program:

```
public class Test {
  public static void main(String args[]) {
    C1 object = new C1();
    object.f(4, 3);
  }
}
class C1 {
  public void f(int xx, final int yy) {
    int a = xx + yy;
    final int b = xx - yy;
    class C2 {
      public void g() {
        System.out.print("a = " + a);
        System.out.print(", b = " + b);
      }
    }
    C2 obiectDoi = new C2();
    obiectDoi.g();
  }
}
```

a) Programul este corect şi va afişa la execuţie: a = 7, b = 1.
b) Programul este corect şi va afişa la execuţie: a = 4, b = 3.
c) Va apărea eroare la compilare, deoarece din metoda `g()` nu putem accesa variabila locală a din metoda `f()`.
d) Va apărea eroare la compilare, deoarece clasa `C2` nu poate fi definită în metoda `f()` din clasa `C1`.
e) Va apărea eroare la compilare, deoarece nu se creează în clasa `Test` un obiect de tip `C1.C2`.

> Răspuns: A

Întrebarea 3.7.24.

Verificaţi corectitudinea următorului program Java:

```
public class Exemplu {
  int v1[] = {0}, v2[] = {1};
  public static void main(String args[]) {
    int v1[] = {2}, v2[] = {3};
    System.out.print(v1[0] + " " + v2[0] + " ");
    f(v1, v2);
    System.out.println(v1[0] + " " + v2[0]);
  }
  public static void f(int v1[], int v2[]) {
    int w[] = {4};
    v1 = w;
    v2[0] = w[0];
    System.out.print(v1[0] + " " + v2[0] + " ");
  }
}
```

a) Programul este corect şi va afişa la execuţie: 0 1 4 4 0 1.
b) Programul este corect şi va afişa la execuţie: 2 3 4 4 0 1.
c) Programul este corect şi va afişa la execuţie: 2 3 4 4 2 4.
d) Programul este corect şi va afişa la execuţie: 2 3 4 4 4 4.
e) Va apărea eroare la compilare, deoarece atributele `v1` şi `v2` sunt redefinite ca variabile locale în metoda `main()`.

> Răspuns: C

Întrebarea 3.7.25.

Ce puteţi spune despre următorul program Java (liniile sunt numerotate)?

```
class C1 { int x = 1; }
class C2 extends C1 { int y = 2; }
public class TestCompilare {
  public static void main(String[] args) {
    C1 obiectUnu = new C1();
    System.out.print(obiectUnu.x + " ");
    obiectUnu = new C2();
    System.out.print(obiectUnu.x + " ");
    C2 obiectDoi = (C2) obiectUnu;
    System.out.print(obiectDoi.x + " ");
  }
}
```

a) Programul este corect şi va afişa la execuţie: 1 1 1.
b) Va apărea eroare la compilarela linia 7, deoarece `obiectUnu` este deja definit la linia 5.
c) Va apărea eroare la compilare la linia 9, deoarece `obiectUnu` este de tip `C1` definit la linia 5.
d) Vor apărea erori de compilare la liniile 5, 7 şi 9, deoarece într-o metodă statică nu putem crea variabile obiect.
e) Va apărea eroare la execuţie, aruncând excepţia `ClassCastException`.

> Răspuns: A

Întrebarea 3.7.26.

Stabiliţi care afirmaţii sunt adevărate pentru programul Java din cele de mai jos (liniile sunt numerotate):

```
public class TestTablouri {
  public static void main(String[] args) {
    int[] a = { 2, 6 };
    Object b = a;
    System.out.print(b[0] + " ");
    byte[] c = new byte[2];
    c = a;
    System.out.print(c[1] + " ");
  }
}
```

a) Programul este corect şi va afişa la execuţie: 2 6.
b) Va apărea eroare la compilare doar la linia 4, deoarece `b` necesită o conversie explicită.
c) Va apărea eroare la compilare doar la linia 5, deoarece `b[0]` nu poate fi accesat.
d) Vor apărea erori de compilarela linia 5, deoarece `b[0]` nu poate fi accesat şi la linia 7, deoarece tipurile componentelor tablourilor `a` şi `c` nu sunt compatibile.
e) Va apărea eroare la execuţie din cauza liniei 5, aruncându-se excepţia `ClassCastException`.

> Răspuns: D

Întrebarea 3.7.27.

Indicaţi dacă program Java de mai jos este corect sau stabiliţi natura erorilor (liniile sunt numerotate):

```
1. class C1 { int x = 1; }
2. class C2 extends C1 { int y = 2; }
3. interface I1 { int x = 3; void f(int x); }
4. class C3 extends C1 implements I1 {
5.  int x = 4;
6.  public void f(int x) { this.x = x; }
7. }
8. public class TestConversii {
9.   public static void main(String[] args) {
10.    C3 obiectUnu = new C3();
11.    I1 obiectDoi = obiectUnu;
12.    System.out.print(obiectDoi.x);
13.    C2[] obiectTrei = new C2[3];
14.    System.out.print(obiectTrei[0].x + " ");
15.    C1[] obiectPatru = obiectTrei;
16.    System.out.print(obiectPatru[0].x + " ");
17.    obiectTrei = (C2[]) obiectPatru;
18.  }
19.}
```

a) Programul este corect şi va afişa la execuţie: 3 2 1.
b) Programul este corect la compilare şi va afişa la execuţie 3, urmat de aruncarea excepţiei `NullPointerException` la linia 14.
c) Programul este corect la compilare şi va afişa la execuţie 3, urmat de aruncarea excepţiei `NullPointerException` la linia 16.
d) Va apărea eroare la compilare doar la linia 15, deoarece tipurile componentelor tablourilor `obiectTrei` şi `obiectPatru` nu sunt compatibile.
e) Va apărea eroare la execuţie la linia 17, aruncându-se excepţia `ClassCastException`.

> Răspuns: B

Întrebarea 3.7.28.

Ce puteţi spune despre următorul program Java?

```
class C1 { int x = 1; }
interface I1 { int x = 2; void f(int x); }
class C2 extends C1 implements I1 {
  int x = 3;
  public void f(int x) { this.x = x; }
}
public class Test {
  public static void main(String[] args) {
    C1 obiectUnu = new C2();
    System.out.print(obiectUnu.x);
    I1 obiectDoi = (C2) obiectUnu;
    System.out.println(" " + obiectDoi.x);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 1 2.
b) Programul este corect şi va afişa la execuţie: 3 2.
c) Programul este corect şi va afişa la execuţie: 3 3.
d) Va apărea eroare de compilare la crearea lui `obiectDoi`.
e) Va apărea eroare la execuţie la afişarea lui `obiectDoi.x`, aruncându-se excepţia `ClassCastException`.

> Răspuns: A

Întrebarea 3.7.29.

Verificaţi dacă programul de mai jos este corect:

```
interface I1 { int x = 1; void f(int x); }
class C1 implements I1 {
  int x = 2;
  public void f(int x) { this.x = x; }
}
public class Test {
  public static void main(String[] args) {
    C1 obiectUnu = new C1();
    System.out.print(obiectUnu.x);
    I1 obiectDoi = obiectUnu;
    System.out.println(" " + obiectDoi.x);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 2 1.
b) Programul este corect şi va afişa la execuţie: 2 2.
c) Va apărea eroare de compilare la crearea lui `obiectDoi`, fiind necesară o conversie explicită de forma (`C2`).
d) Va apărea eroare de compilare la crearea lui `obiectDoi`, deoarece atributul `x` se suprascrie.
e) Va apărea eroare la execuţie la afişarea lui `obiectDoi.x`, aruncându-se excepţia `ClassCastException`.

> Răspuns: A

Întrebarea 3.7.30.

Ce puteţi afirma despre următorul program Java?

```
class C1 {
  int x = 1;
  void f(int x) {
    System.out.print("Unu ");
    this.x = x;
  }
}
interface I1 {
  int x = 2;
  void f(int x);
}
class C2 extends C1 implements I1 {
  int x = 3;
  public void f(int x) {
    System.out.print("Doi ");
    this.x = x;
  }
}
public class Test {
  public static void main(String[] args) {
    C1 obiectUnu = new C2();
    System.out.print(obiectUnu.x + " ");
    obiectUnu.f(4);
    I1 obiectDoi = (C2) obiectUnu;
    obiectDoi.f(5);
    System.out.print(obiectDoi.x);
  }
```

```
}
```

a) Programul este corect şi va afişa la execuţie: 1 Unu Doi 2.
b) Programul este corect şi va afişa la execuţie: 1 Unu Doi 3.
c) Programul este corect şi va afişa la execuţie: 1 Doi Doi 2.
d) Programul este corect şi va afişa la execuţie: 3 Doi Doi 2.
e) Programul este corect şi va afişa la execuţie: 3 Doi Doi 3.
f) Va apărea eroare de compilare la crearea lui `obiectDoi`, deoarece atributul `x` se suprascrie.
g) Va apărea eroare la execuţie la afişarea lui `obiectDoi.x`, aruncându-se excepţia `ClassCastException`.

> Răspuns: C

Întrebarea 3.7.31.

Indicaţi dacă programul de mai jos este corect şi, în caz afirmativ, ce va apărea pe ecran:

```
class C1 {
  int x = 1;
  void f(int x) {
    System.out.print("1 ");
    this.x = x;
  }
}
class C2 extends C1 {
  int x = 2;
  public void f(int x) {
    System.out.print("2 ");
    this.x = x;
  }
}
class C3 extends C1 {
  int x = 3;
  public void f(int x) {
    System.out.print("3 ");
    this.x = x;
  }
}
public class Test {
  public static void main(String[] args) {
    C1 obiectUnu = new C2();
    C1 obiectDoi = new C3();
    System.out.print(obiectUnu.x + " " + obiectDoi.x + " ");
    obiectUnu.f(4);
    obiectDoi.f(5);
    System.out.print(obiectUnu.x + " " + ((C2) obiectUnu).x + " ");
    System.out.print(obiectDoi.x + " " + ((C3) obiectDoi).x);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 1 1 1 1 1 4 1 5;
b) Programul este corect şi va afişa la execuţie: 1 1 2 3 1 4 1 5;
c) Programul este corect şi va afişa la execuţie: 2 3 2 3 2 4 3 5;
d) Programul este corect şi va afişa la execuţie: 2 3 2 3 4 4 5 5;

e) Vor apărea erori de compilare la crearea instanţelor `obiectUnu` şi `obiectDoi`, deoarece atributul `x` se suprascrie.
f) Va apărea eroare la execuţie la afişarea lui `obiectUnu.x`, aruncându-se excepţia `ClassCastException`.

> Răspuns: B

Întrebarea 3.7.32.

Ce puteţi afirma despre programul Java de mai jos?

```
class C1 {
  int x = 1;
  C1(int x) {
    this.x = x;
  }
}
class C2 extends C1 {
  int x = 3, y = 4;
  C2(int x, int y) {
    super(x);
    this.y = y;
  }
}
public class Test {
  public static void main(String[] args) {
    C1[] tablouUnu = new C2[3];
    for (int i = 0; i < tablouUnu.length; i++)
      tablouUnu[i] = new C2(i * 2, i * 3);
    C2[] tablouDoi = (C2[]) tablouUnu;
    for (int i = 0; i < tablouUnu.length; i++)
      System.out.print(tablouDoi[1].x + " " + tablouDoi[i].y + " ");
  }
}
```

a) Programul este corect şi va afişa la execuţie: 1 0 1 3 1 6.
b) Programul este corect şi va afişa la execuţie: 1 4 1 4 1 4.
c) Programul este corect şi va afişa la execuţie: 3 0 3 3 3 6.
d) Programul este corect şi va afişa la execuţie: 3 4 3 4 3 4.
e) Vor apărea erori de compilare la crearea instanţelor `tablouUnu` şi `tablouDoi`, deoarece atributul `x` se suprascrie.
f) Va apărea eroare la execuţie la afişarealui `tablouUnu.x`, aruncându-se excepţia `ClassCastException`.

> Răspuns: C

Întrebarea 3.7.33

Verificaţi corectitudinea programului următor şi specificaţi ce se va afişa:

```
class C1 {
  int x = 1;
  C1(int x) { this.x = x; }
}
class C2 extends C1 {
```

```
  int x = 2, y = 3;
  C2(int x) { super(x); }
  C2(int x, int y) { super(x); this.y = y; }
}
public class ApelExplicitConstructor {
  public static void main(String[] args) {
    C1 obiectUnu = new C1(4);
    C2 obiectDoi = new C2(5);
    C2 obiectTrei = new C2(6, 7);
    System.out.print(obiectUnu.x + " ");
    System.out.print(obiectDoi.x + " " + obiectDoi.y + " ");
    System.out.print(obiectTrei.x + " " + obiectTrei.y);
  }
}
```

a) Programul este corect şi va afişa la execuţie: 4 1 2 1 7.
b) Programul este corect şi va afişa la execuţie: 4 1 3 6 7.
c) Programul este corect şi va afişa la execuţie: 4 2 3 2 7.
d) Programul este corect şi va afişa la execuţie: 4 5 3 6 7.
e) Vor apărea erori de compilare la crearea instanţelor `obiectDoi` şi `obiectTrei`, deoarece atributul `x` se suprascrie.
f) Va apărea eroare la execuţie la afişarea lui `ObiectDoi.x`, aruncându-se excepţia `ClassCastException`.

> Răspuns: C