Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos:

```
abstract class A
{
    public int proc(A p)
    {
        return 98;
    }
}

class B extends A
{
    public int proc(A p)
    {
        return 17;
    }
}

class C extends A
{
    public int proc(C p)
    {
        return 65;
    }
}

public class Main
{
    public static void main(String argv[])
    {
        C x = new C();
        A y = new B();
        C z = new C();
        System.out.println(y.proc(x) + z.proc(x)); //*
    }
}
```

Răspuns: 17 + 65 = 82

Question **3**

Not yet answered

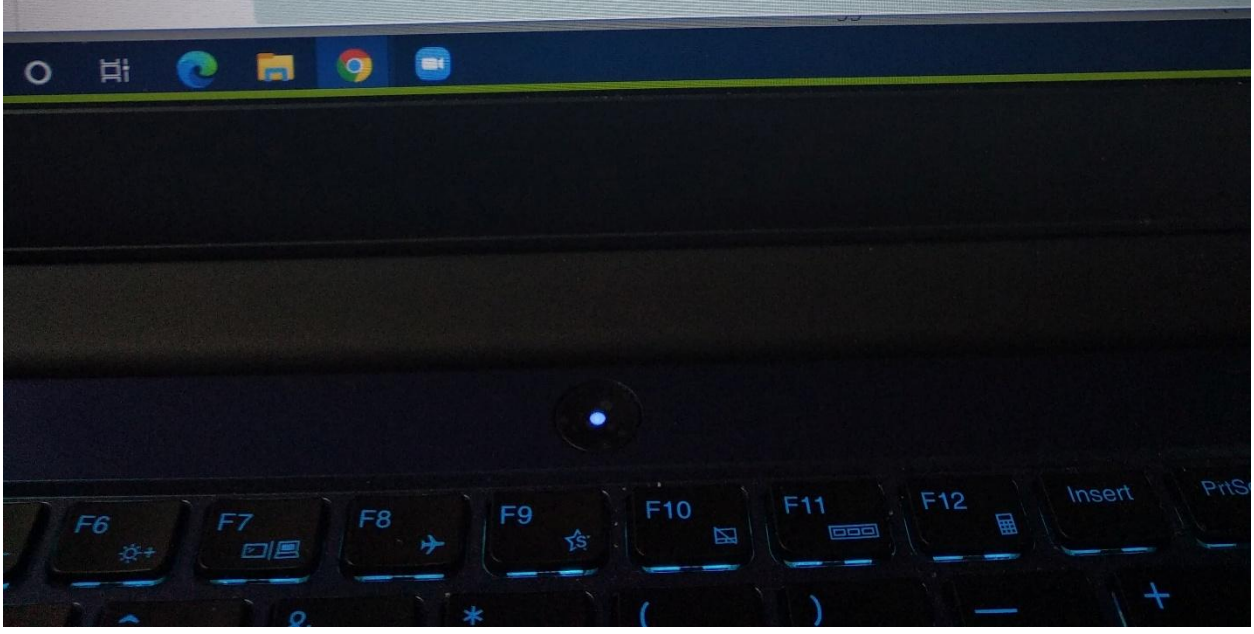
Marked out of 3.00

Flag question

**Ce valoare întreagă se va tipării la linia marcată cu //* în urma execuției programulu**

```
abstract class A {
    public int proc(A p) { return 74; }
}
class B extends A {
    public int proc(A p) { return 33;}
}
class C extends A {
    public int proc(C p) { return 71; }
}
public class Main {
    public static void main(String argv[]) {
        C x = new C();
        A y = new B();
        C z = new C();
        System.out.println(y.proc(x) + z.proc(x)); //*
    }
}
```

Answer:



e Chrome

56&cmid=88563&page=1

on 2

t

ed

d out of

question

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de

```
class Pair {
    private int v;
    private static int t[];
    public Pair(int x, int[] y) { v = x; t = y; }
    public void setVT(int x, int[] y) { v = x; t = y; }
    public void setVIT(int x, int i, int y) { v = x; t[i] = y; }
    public int getV() { return v; }
    public int[] getT() { return t; }
}
class Main {
    public static Pair call(Pair p, Pair q) {
        p.setVT(17, q.getT());
        p = q;
        p.setVIT(72, 1, 21);
        int tz[] = {89, 41};
        return new Pair(94, tz);
    }
    public static void main(String[] args) {
        int ta[] = {0, 13};
        int tb[] = {45, 99};
        Pair a = new Pair(66, ta);
        Pair b = new Pair(55, tb);
        Pair c = b;
        b = call(a, c);
        System.out.println(a.getT()[0]); //*
    }
}
```

Answer:

Not yet answered

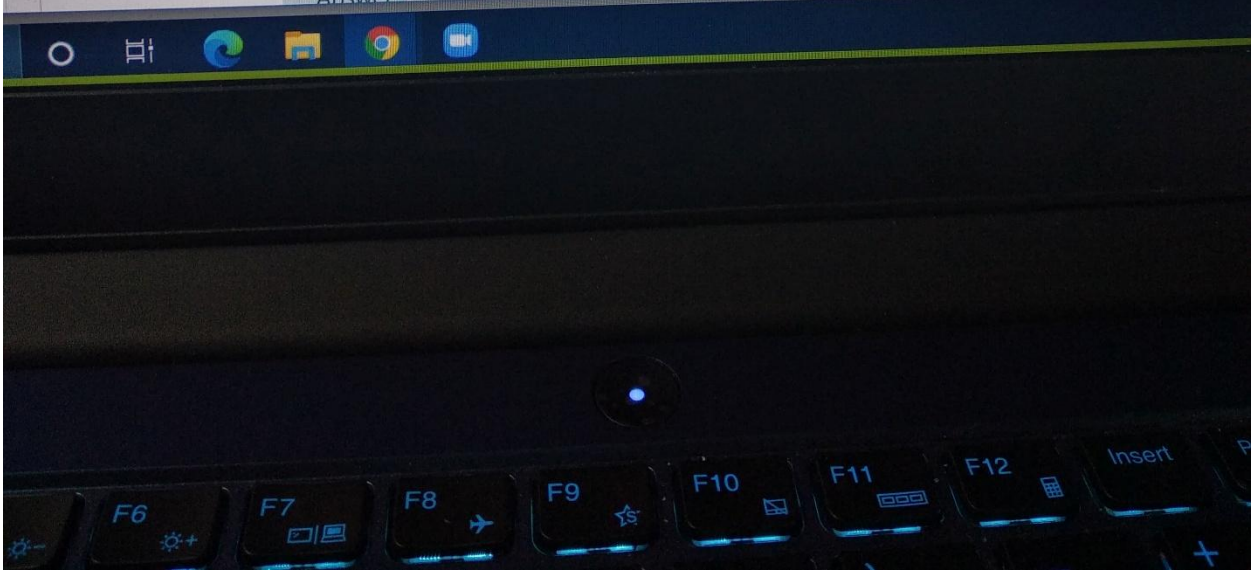
Marked out of 3.00

Flag question

**Ce valoare întreagă se va tipării la linia marcată cu //* în urma execuției programului**

```
class Ex extends Exception {}
class Main {
    public static void main(String argv[]) throws Ex {
        int i = 0;
        int z = 0;
        while (i < 2) {
            try {
                int k = 0;
                while (k < 3) {
                    k++;
                    z = z + 2;
                    if (i == 0) throw new Ex();
                }
                z = z + 2;
            } catch (Ex e) {
                System.out.println(e);
            } finally {
                z = z + 1;
            }
            i++;
        }
        System.out.println(z); //*
    }
}
```

Answer:



# e Obiecte

Question 4

Not yet answered

Marked out of .00

Flag question

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției prog

```
abstract class A {
    public int proc(B p) { return 25; }
}
class B extends A {
    public int proc(C p) { return 96;}
}
class C extends B {
    public int proc(C p) { return 99; }
}
public class Main {
    public static void main(String argv[]) {
        A x = new C();
        C y = new C();
        C z = new C();
        System.out.println(x.proc(z) + y.proc(z)); //*
    }
}
```

Answer:

evious page

GET IT ON

Download on the

Question 3

Not yet answered

Marked out of 3.00

Flag question

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției pro

```
class ExA extends Exception {}
class ExB extends ExA {}
class Main {
    public static int make(int a) throws ExA {
        for (int k = 0; k < 3; k++) {
            a = a + 2;
            if (a == 3) throw new ExB();
        }
        return a;
    }
    public static void main(String argv[]) throws ExA {
        int x = 0;
        for (int i = 0; i < 2; i++) {
            try {
                x++;
                x = make(x);
                x++;
            } catch (ExB e) {
                System.out.println(e);
            } finally {
                x++;
            }
        }
        System.out.println(x); //*
    }
}
```

Answer:

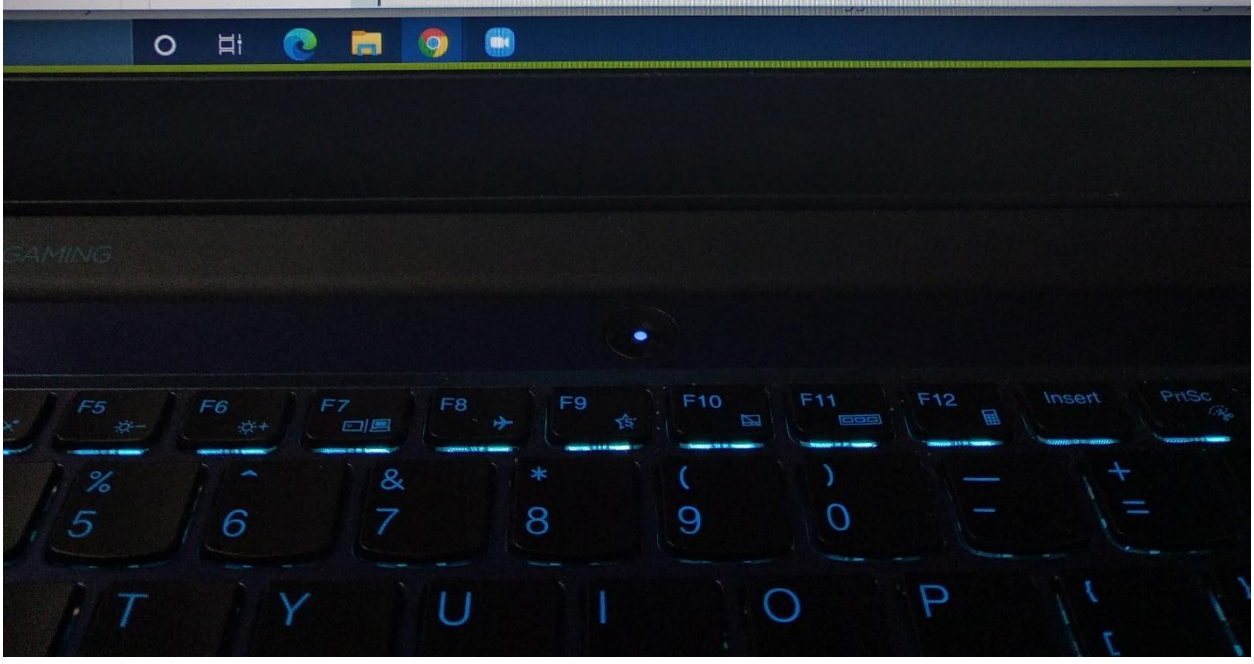
Not yet answered

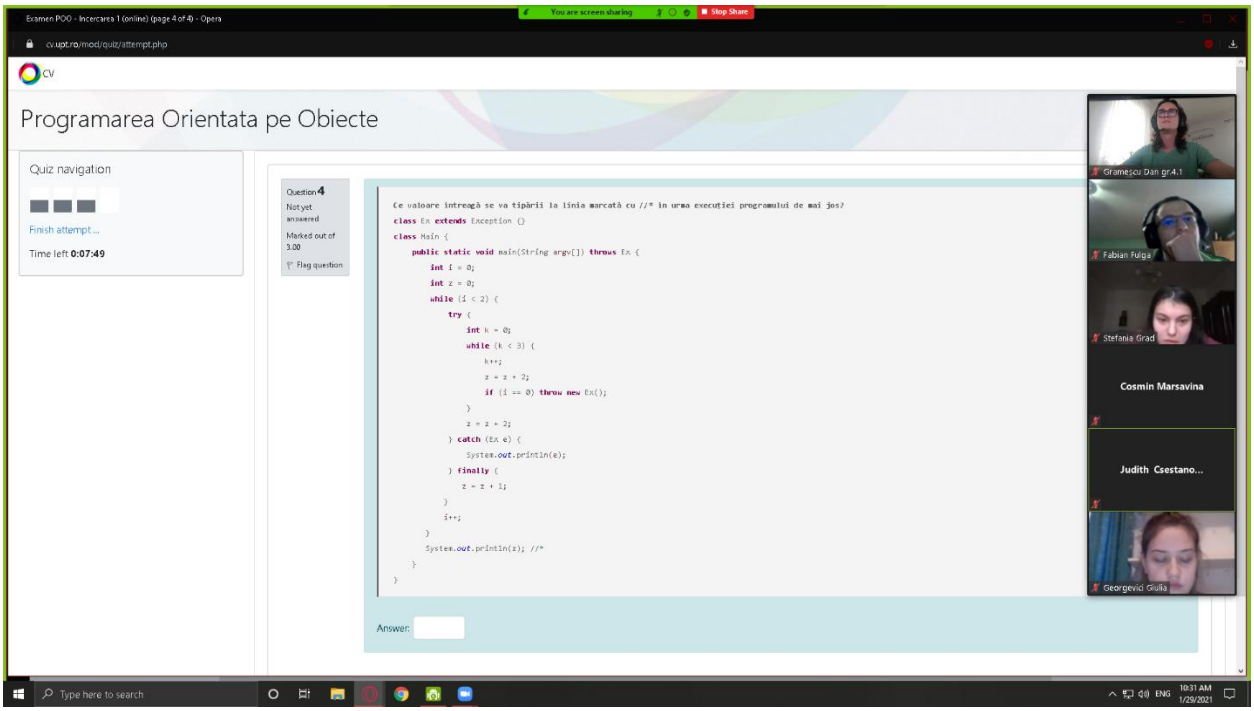
Marked out of 3.00

Flag question

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai

```
class Pair {
    private int v;
    private int t[];
    public Pair(int x, int [] y) { v = x; t = y; }
    public void setVT(int x, int[] y) { v = x; t = y; }
    public void setVIT(int x, int i, int y) { v = x; t[i] = y; }
    public int getV() { return v; }
    public int[] getT() { return t; }
}
class Main {
    public static Pair call(Pair p, Pair q) {
        q.setVT(100, p.getT());
        q = p;
        q.setVIT(53, 1, 34);
        int tz[] = {88, 81};
        return new Pair(20, tz);
    }
    public static void main(String[] args) {
        int ta[] = {71, 59};
        int tb[] = {11, 61};
        Pair a = new Pair(19, ta);
        Pair b = new Pair(83, tb);
        Pair c = b;
        b = call(a, b);
        System.out.println(c.getT()[1]); //*
    }
}
```






Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*

Timp rămas 0:21:08

```
//A.java
package p1;
public class A {
    private void m() {}
    void n() {}
    protected void p() {}
    public void q() {}
}
```

```
//B.java
package p2;
import p1.A;
public class B extends A {
    private void r() {}
    void s() {}
    protected void t() {}
    public void k() {}
}
```

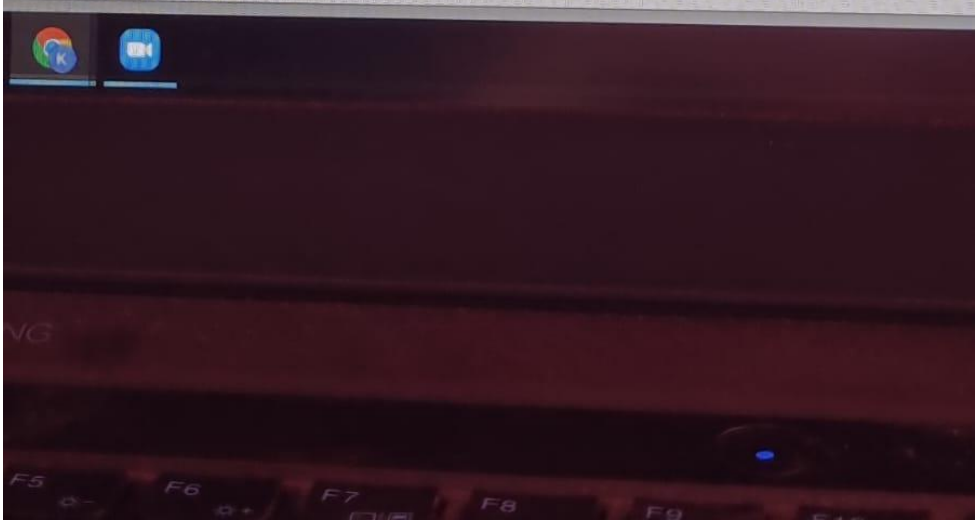
```
//C.java
package p2;
import p1.A;
public class C {
    private void z() {}
    public void v(A a, B b, C c) {
        /*AICI*/
    }
}
```

```
    public void v(A a, B b, C c) {
        /*AICI*/
    }
}
```

Selectați unul sau mai multe:

- [ ] `b.s();`
- [ ] `c.z();`
- [ ] `a.p();`
- [ ] `a.n();`
- [ ] `b.t();`

Pagina precedentă



Considerând următorul cod Java, care din expresiile enumerate mai jos vor avea valoarea de adevăr FALSE când sunt evaluate la poziția indicată cu /*AICI*/?

Timp rămas

```
class A {}
class B extends A {}
class Main {
    public static void main(String argv[]) {
        A a = new B();
        B b = new B();
        Object o = new A();
        System.out.println(/*AICI*/);
    }
}
```

Selectați unul sau mai multe:

- [ ] `A.class.equals(a.getClass())`
- [ ] `A.class.isInstance(b)`
- [ ] `o.getClass().equals(Object.class)`
- [ ] `a.getClass().equals(b.getClass())`
- [ ] `B.class.isInstance(a)`

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?

```
class A {
    public int m(A p) { return 8; }
    public int n(A p) { return 7; }
}
class B extends A {
    public int m(B p) { return 2;}
    public int n(A p) { return 44; }
}
class C extends A {
    public int m(A p) { return 4; }
    public int n(A p) { return 4; }
}
class D extends A {
    public int m(A p) { return 11; }
    public int n(D p) { return 30; }
}
class Main {
    public static void main(String argv[]) {
        A x = new C();
        A y = new D();
        D z = new D();
        System.out.println(x.n(z) + y.m(z));//*
    }
}
```

Răspuns:

**Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*AICI*/?**

```
interface I {}
interface J {}
interface IJ extends I,J {}
class A implements I {}
class B extends A implements J {}
class C extends A {}
class D implements IJ {}
class Main {
    public void doSomething(I i, J j, IJ ij) {
        A a = new A();
        B b = new B();
        C c = new C();
        D d = new D();
        /*AICI*/
    }
}
```

Selectați unul sau mai multe:

- [ ] `j = c;`
- [ ] `b = d;`
- [ ] `i = c;`
- [ ] `ij = j;`
- [ ] `i = ij;`

Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*AICI*/?

```
interface I {
    public void g();
}
interface J extends I {
    public void h();
}
abstract class A implements I {
    public void m() {}
}
class B extends A implements J {
    public void p() {}
    public void g() {}
    public void h() {}
}
class Main {
    public static void main(String argv[]) {
        J j = new B();
        A a = new B();
        /*AICI*/
    }
}
```

Selectați unul sau mai multe:

- [ ] `j.g();`
- [ ] `a.p();`
- [ ] `a.g();`
- [ ] `j.m();`
- [ ] `a.h();`

**Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?**

```
class Ex1 extends Exception {}
class Ex2 extends Ex1 {}
class Main {
    public static void m(int x) throws Ex1 {
        if(x == 1) throw new Ex2();
    }
    public static void main(String argv[]) {
        int k = 0;
        int a = 1;
        while (k < 2) {
            try {
                k++;
                m(k);
                a++;
            } catch(Ex2 e) {
                a += 2;
            } catch(Ex1 e) {
                a += 3;
            } finally {
                a++;
            }
        }
        System.out.println(a);//*
    }
}
```

Răspuns:

ebare
primit
ns încă
t din 1,00
rebare cu

**Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?**

```
class A {
    private int x;
    public A(int x) { this.x = x; }
    public void setX(int x) { this.x = x; }
    public int getX() { return x; }
}
class B {
    private A y;
    public B(A y) { this.y = y; }
    public void setY(A y) { this.y = y;}
    public A getY() { return y; }
}
class Main {
    public static void m(B z, B t) {
        A w = new A(4);
        z.setY(w);
        w.setX(7);
        t = new B(new A(3));
    }
    public static void main(String[] args) {
        B a = new B(new A(35));
        B b = new B(new A(33));
        m(a,b);
        System.out.println(a.getY().getX() + b.getY().getX());//*
    }
}
```

Răspuns:

**Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*AICI*/?**

```
class A {
    public A m(int a) {
        return new A();
    }
}
class B extends A {
    public void n(A a) {
        return;
    }
    /*AICI*/
}
```

Selectați unul sau mai multe:

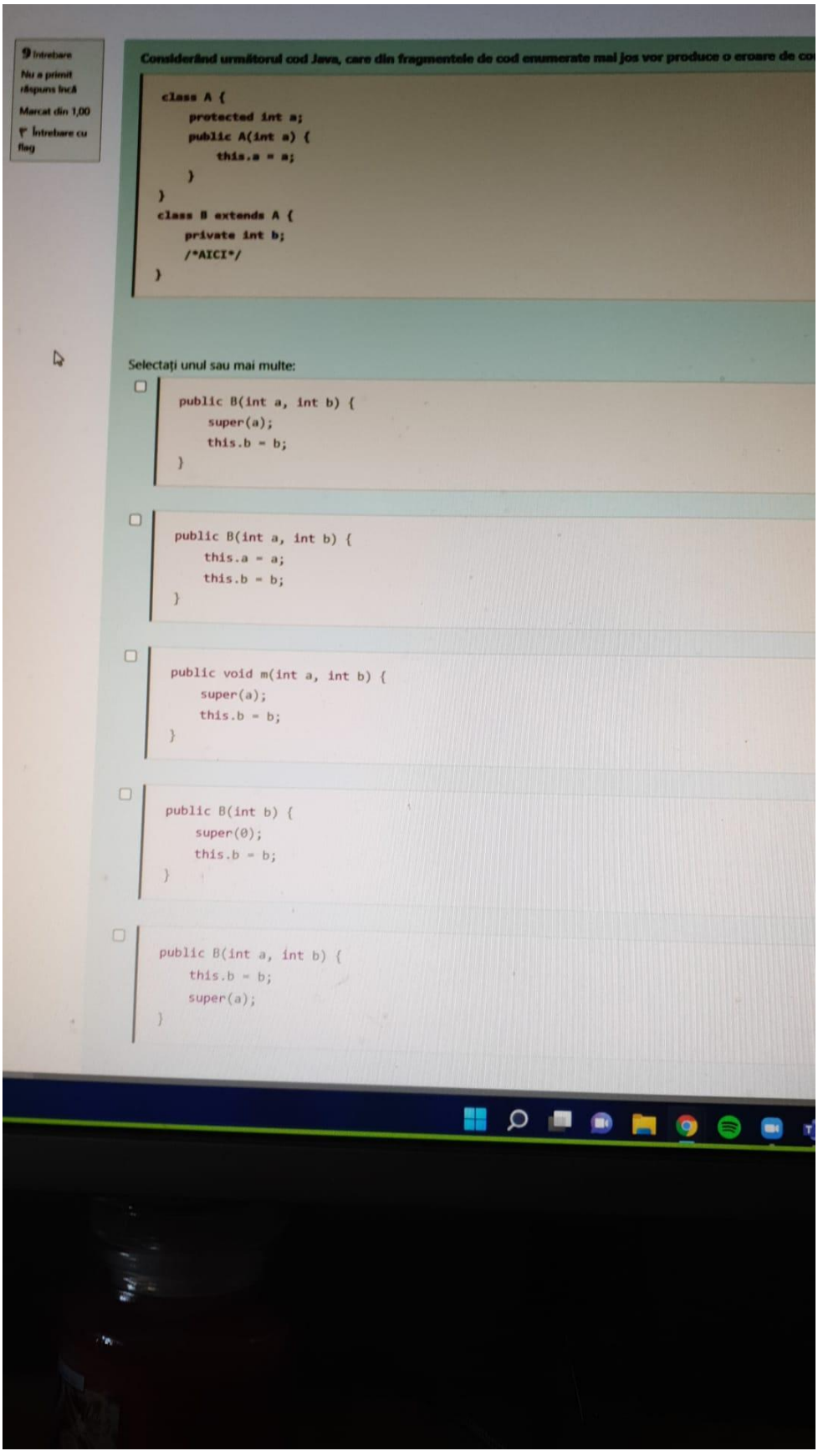
- [ ]
```
public int n(A a) {
   return 0;
}
```

- [ ]
```
public int n(B a) {
    return 0;
}
```

- [ ]
```
public A m(B a) {
    return new B();
}
```

- [ ]
```
public A m(int a) throws Exception {
    throw new Exception("Message");
}
```

- [ ]
```
public A m(int p) {
    return new B();
}
```



9 Întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag

Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de co

```
class A {
    protected int a;
    public A(int a) {
        this.a = a;
    }
}
class B extends A {
    private int b;
    /*AICI*/
}
```

Selectați unul sau mai multe:

```
public B(int a, int b) {
    super(a);
    this.b = b;
}
```

```
public B(int a, int b) {
    this.a = a;
    this.b = b;
}
```

```
public void m(int a, int b) {
    super(a);
    this.b = b;
}
```

```
public B(int b) {
    super(0);
    this.b = b;
}
```

```
public B(int a, int b) {
    this.b = b;
    super(a);
}
```

ă

1,00

e cu

**Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai j**

```
class A {
    private static int x;
    private int y;
    public A(int x1, int y1) { x = x1; y = y1;}
    public void setXY(int x1, int y1) { x = x1; y = y1;}
    public int getX() { return x; }
    public int getY() { return y; }
}
class Main {
    public static void m(A z, A t) {
        z.setXY(3,20);
        z = new A(2,17);
    }
    public static void main(String[] args) {
        A a = new A(9,50);
        A b = new A(7,28);
        m(a,b);
        System.out.println(a.getY() + b.getX());//*
    }
}
```

Răspuns:

Остановить демонстрацию

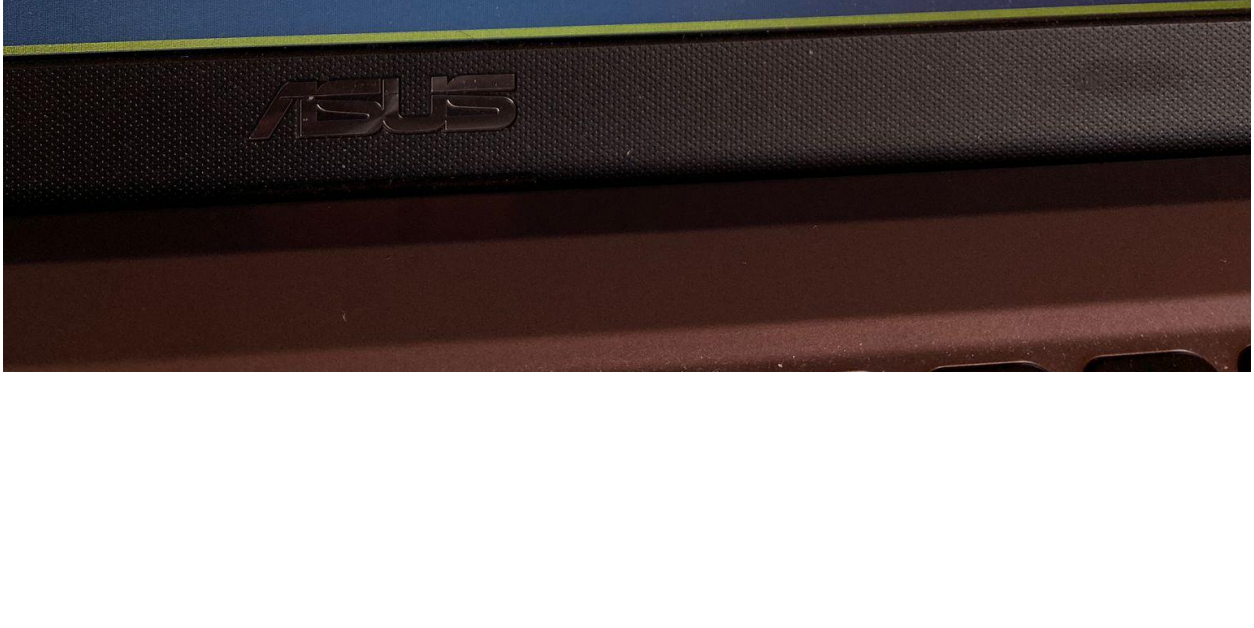
**Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?**

```
class Ex extends RuntimeException {}
class Main {
    public static void m(int x) throws Ex {
        if(x == 2) throw new Ex();
    }
    public static void main(String argv[]) {
        int a = 2;
        try {
            for(int i = 0; i < 4; i++) {
                m(i);
                a = a + 2;
            }
        } catch(Ex e) {
            a++;
        } finally {
            a++;
        }
        System.out.println(a);//*
    }
}
```

Răspuns:

**Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroa sunt inserate la poziția indicată cu /*AICI*/?**

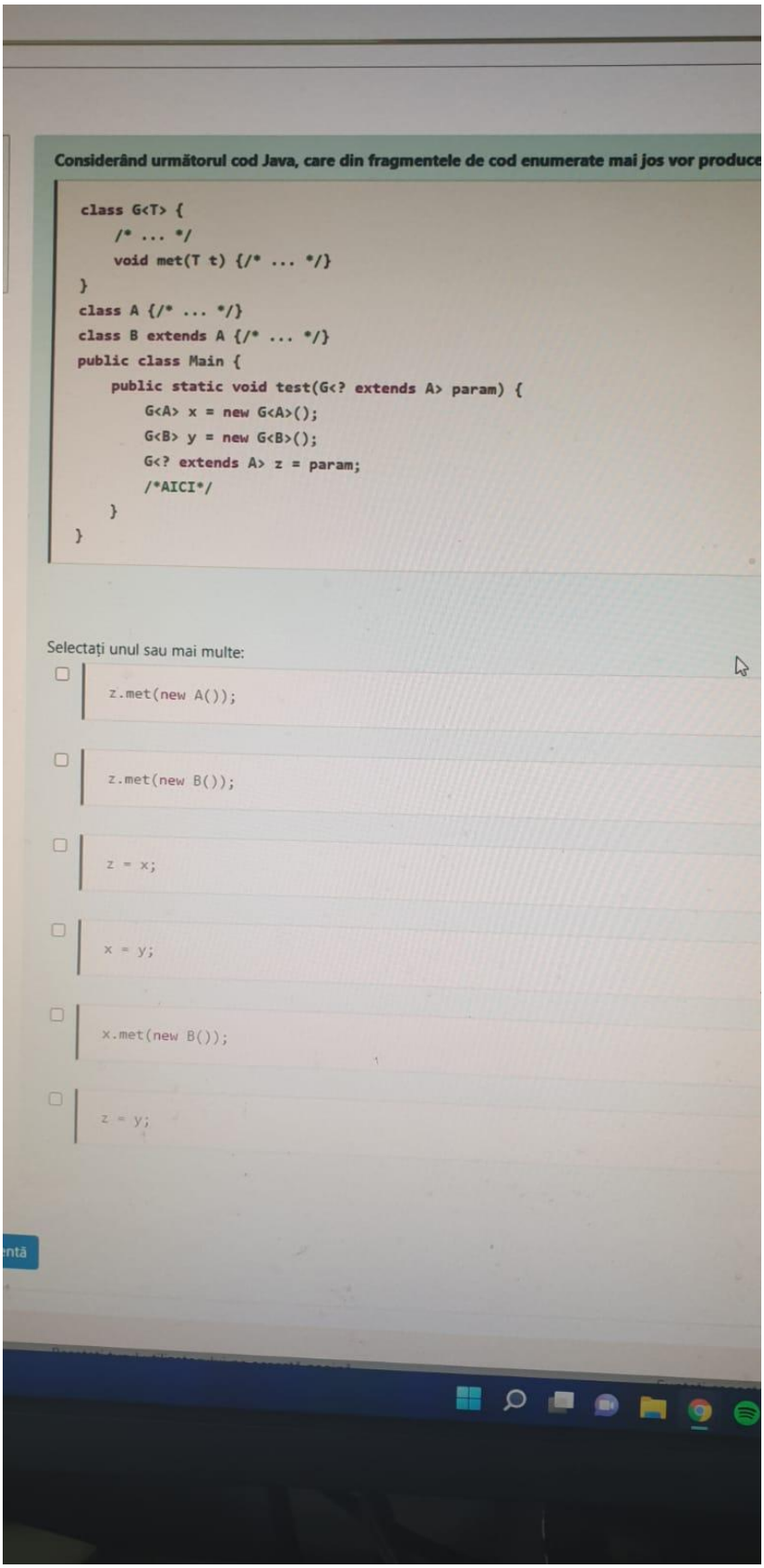
```
abstract class A {
    public void m(int p) {}
    public void r(int p) {}
}
class B extends A {
    public void m(int p) {}
    public void p(int p) {}
    public void q(int p) {}
}
class C extends A {
    public void m(int p) {}
    public void p(int p) {}
    public void r(int p, int t) {}
}
class Main {
    public static void main(String argv[]) {
        A a = new B();
        C c = new C();
        /*AICI*/
    }
}
```



```
    }
}
```

Selectați unul sau mai multe:

- [ ] `c.equals(a);`
- [ ] `c.r(1);`
- [ ] `a.q(1);`
- [ ] `a.p(1);`
- [ ] `a.m(1);`



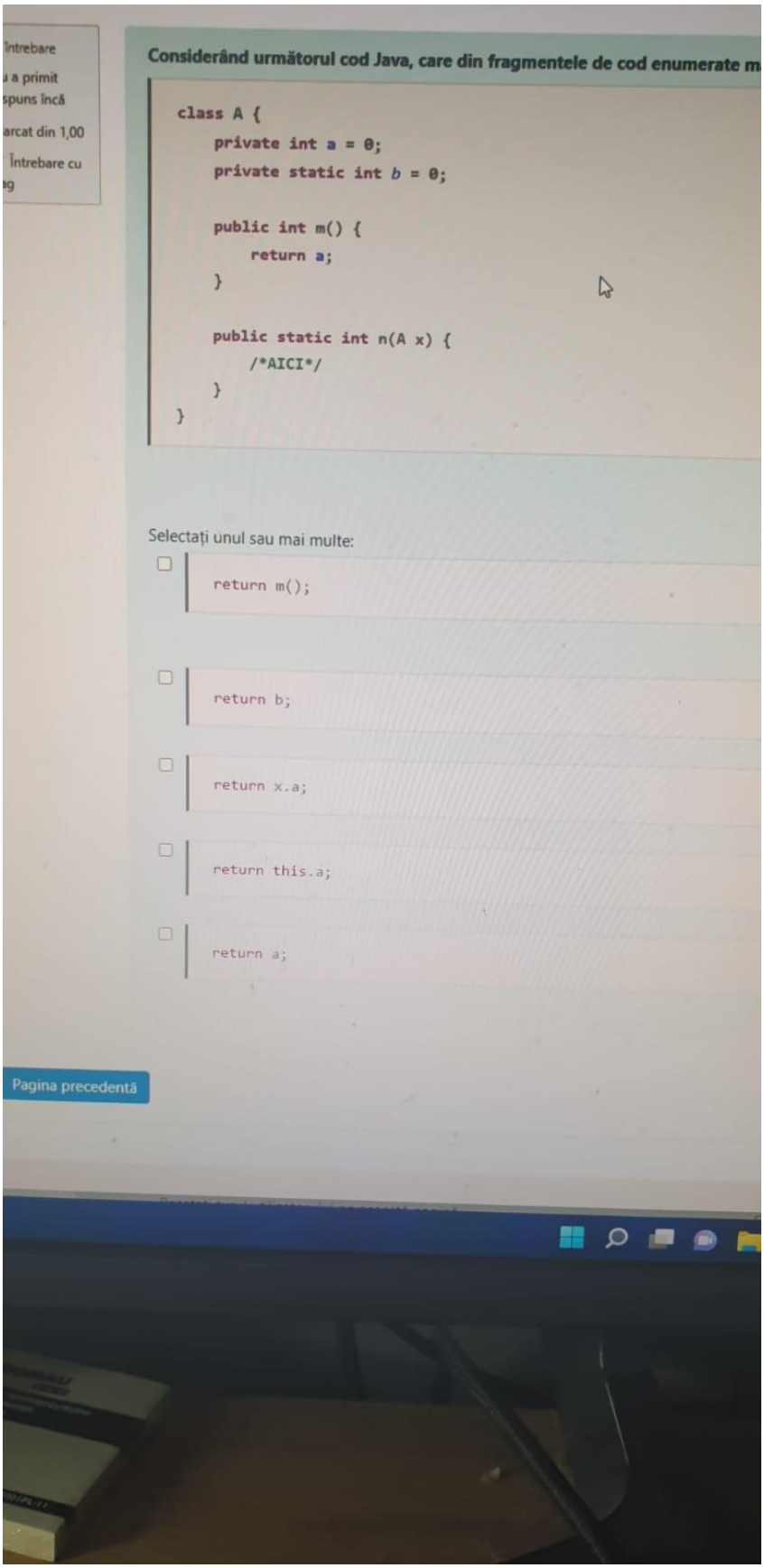
Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce

```
class G<T> {
    /* ... */
    void met(T t) {/* ... */}
}
class A {/* ... */}
class B extends A {/* ... */}
public class Main {
    public static void test(G<? extends A> param) {
        G<A> x = new G<A>();
        G<B> y = new G<B>();
        G<? extends A> z = param;
        /*AICI*/
    }
}
```

Selectați unul sau mai multe:

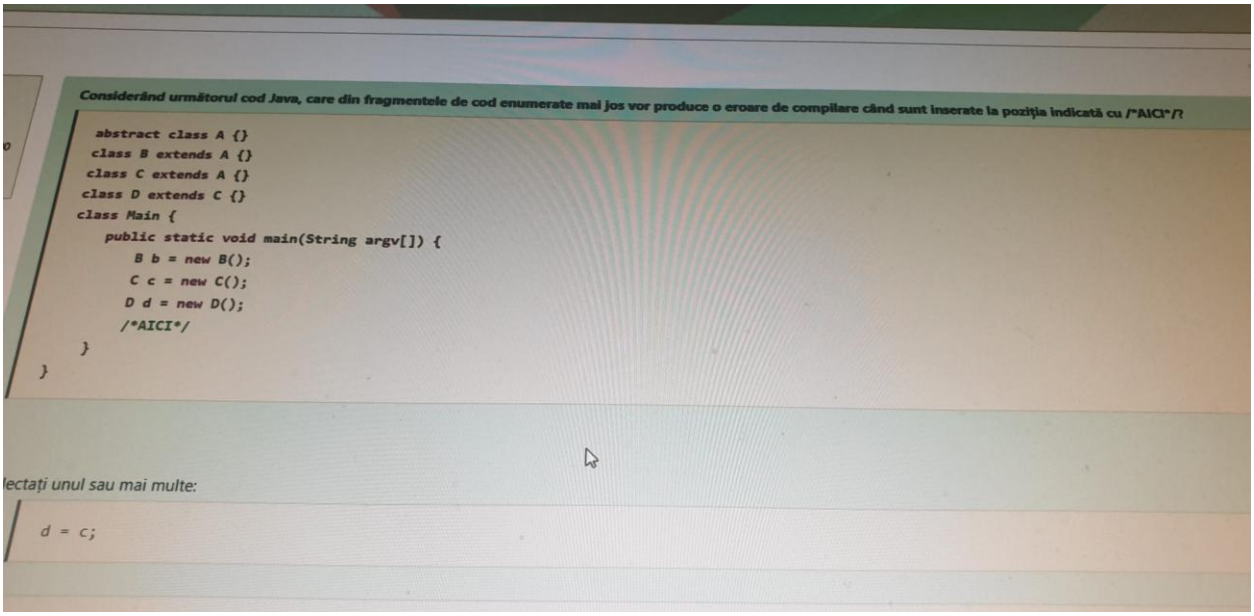
- [ ] `z.met(new A());`
- [ ] `z.met(new B());`
- [ ] `z = x;`
- [ ] `x = y;`
- [ ] `x.met(new B());`
- [ ] `z = y;`

ntă


Întrebare
a primit
spuns încă
arcat din 1,00
Întrebare cu
Considerând următorul cod Java, care din fragmentele de cod enumerate m
class A {
private int a = 0;
private static int b = 0;
public int m() {
return a;
}
public static int n(A x) {
/*AICI*/
}
}
Selectați unul sau mai multe:
return m();
return b;
return x.a;
return this.a;
return a;
Pagina precedentă

**Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?**

```
class A {
    public int m(A p) { return 12; }
    public int n(A p) { return 38; }
}
class B extends A {
    public int m(B p) { return 3;}
    public int n(A p) { return 7; }
}
class C extends A {
    public int m(A p) { return 38; }
    public int n(A p) { return 4; }
}
class D extends A {
    public int m(A p) { return 1; }
    public int n(D p) { return 2; }
}
class Main {
    public static void main(String argv[]) {
        A x = new C();
        A y = new D();
        D z = new D();
        System.out.println(x.m(z) + y.n(z));//*
    }
}
```

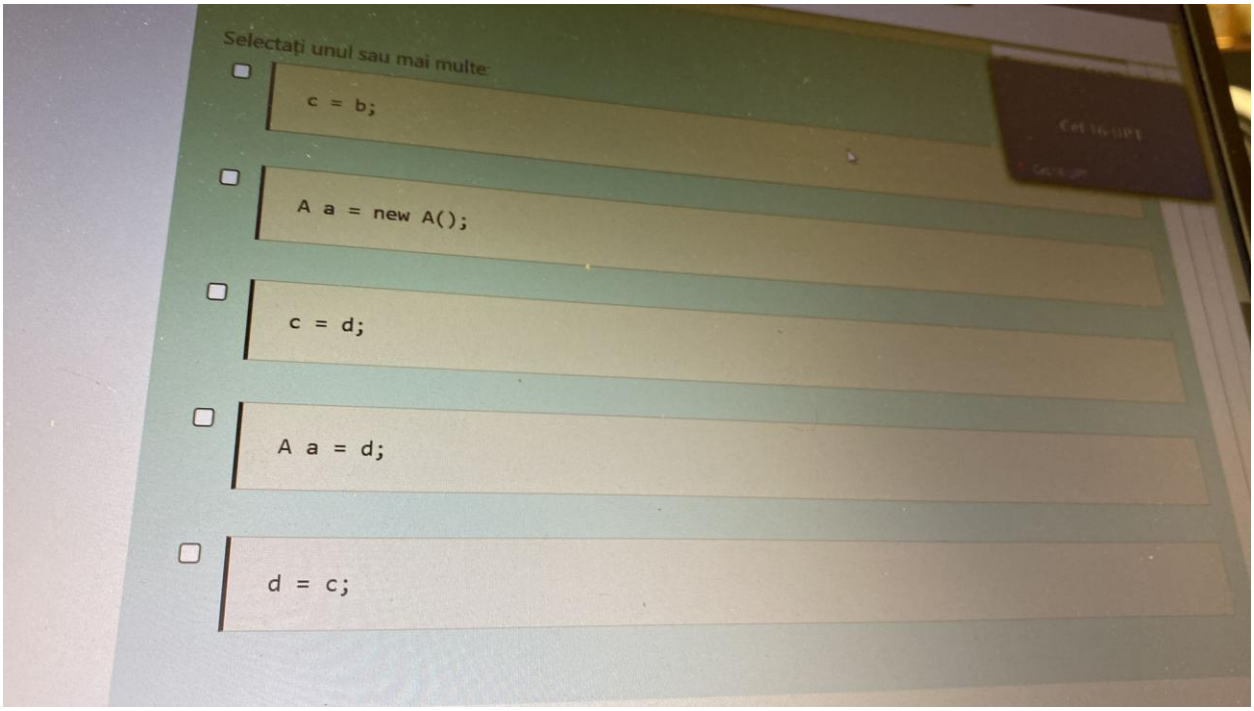


Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*AICI*/?

```
abstract class A {}
class B extends A {}
class C extends A {}
class D extends C {}
class Main {
   public static void main(String argv[]) {
      B b = new B();
      C c = new C();
      D d = new D();
      /*AICI*/
   }
}
```

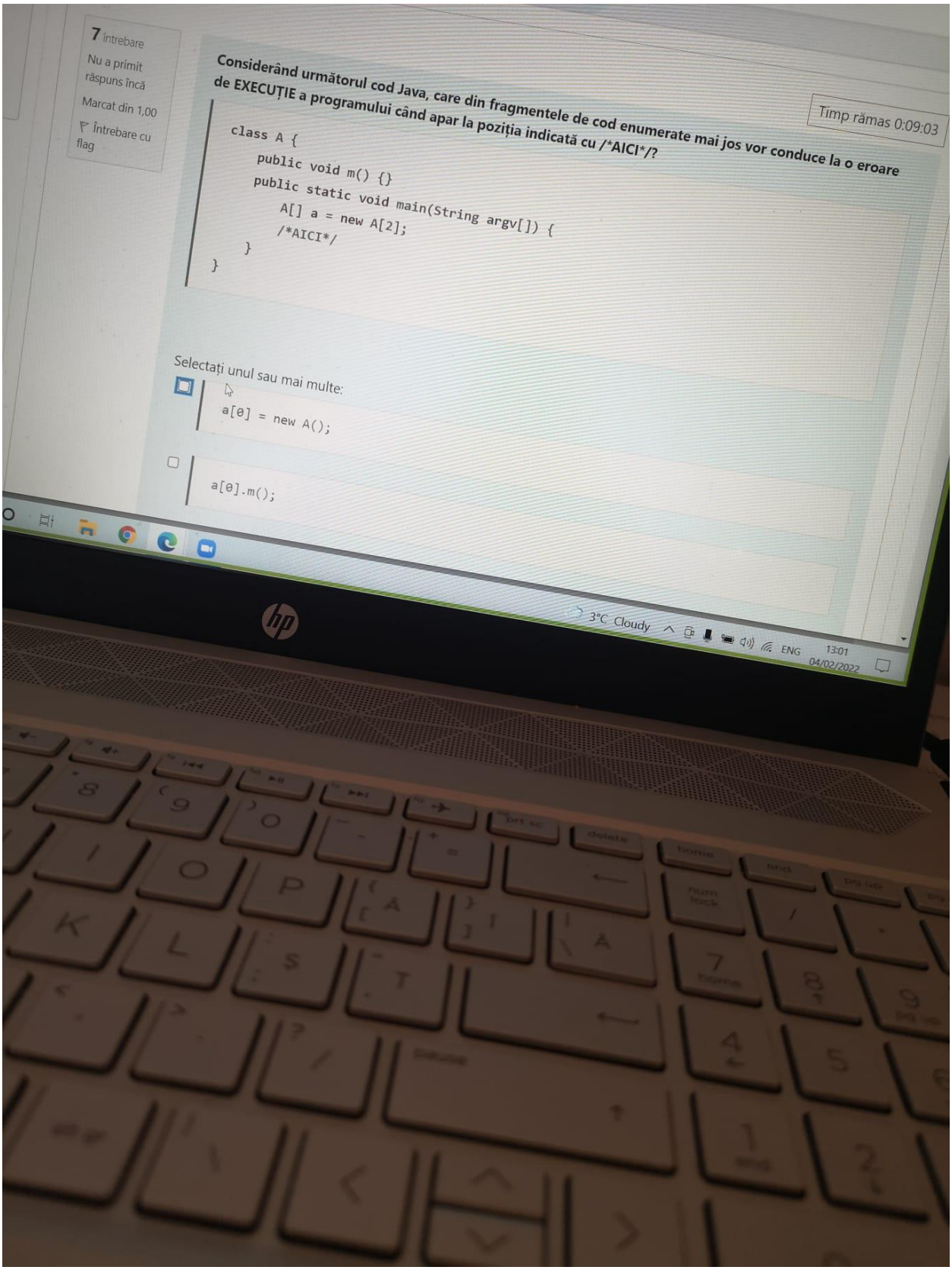
lectați unul sau mai multe:

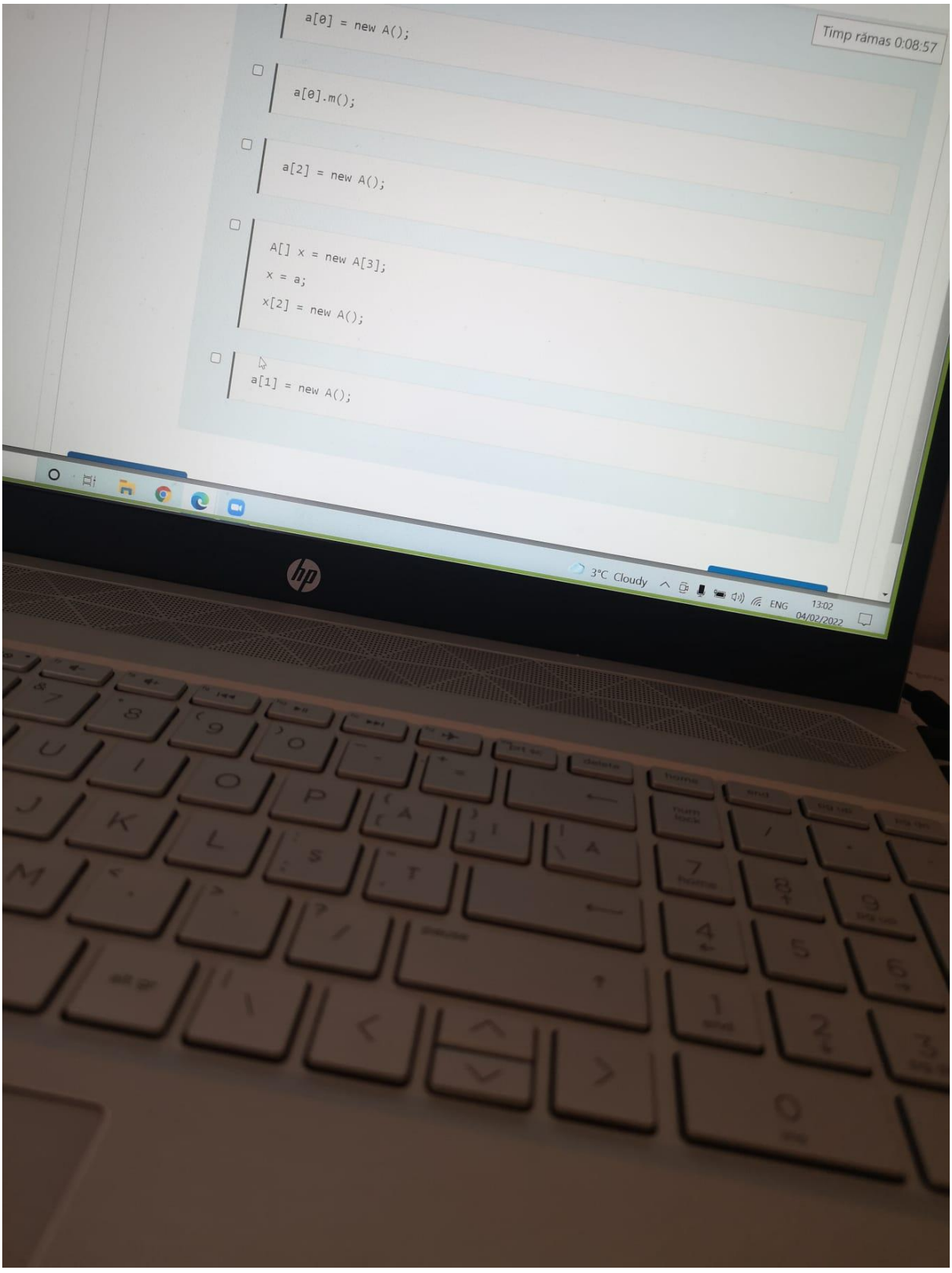
- [ ] `d = c;`

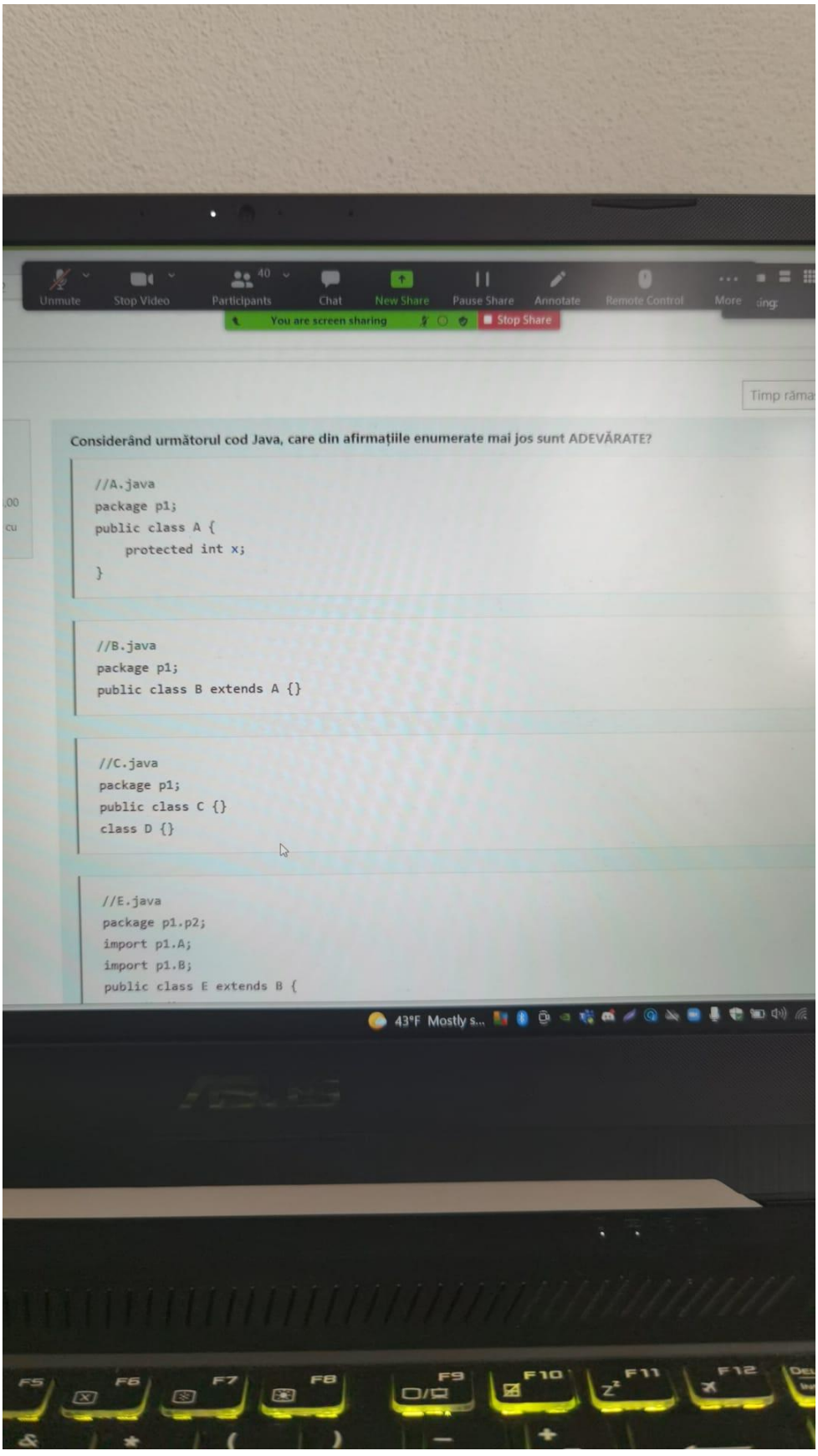


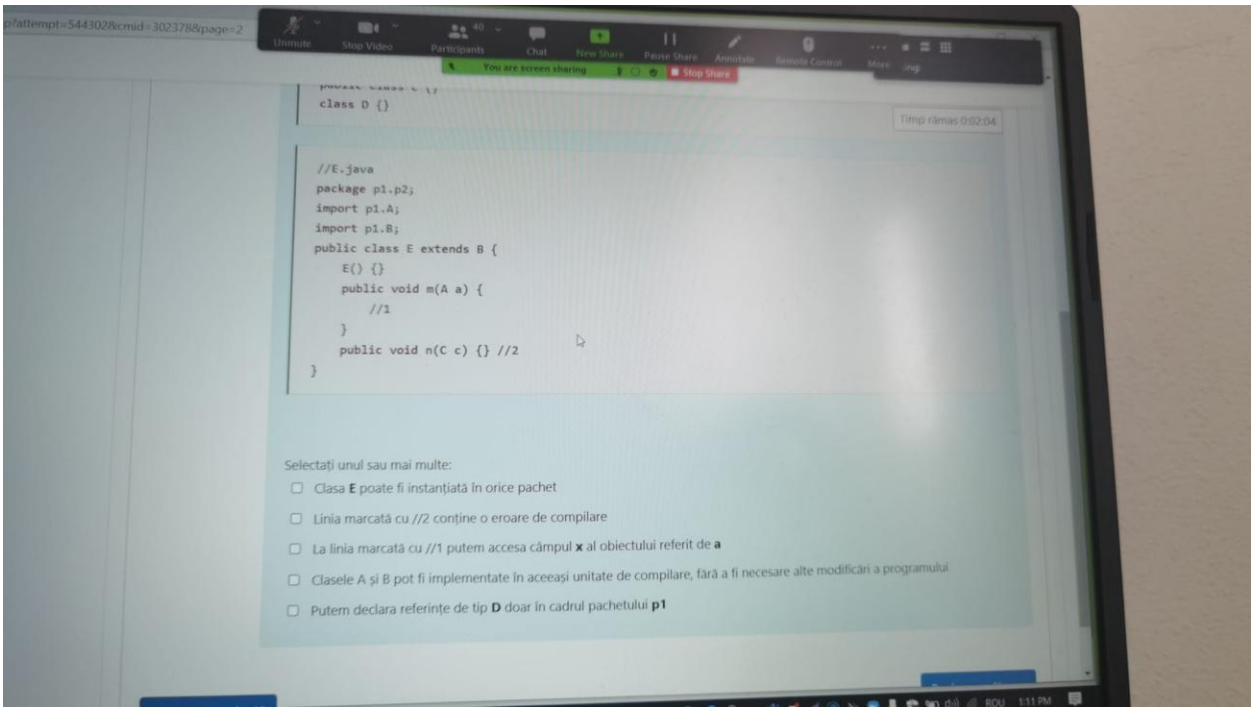
Selectați unul sau mai multe:

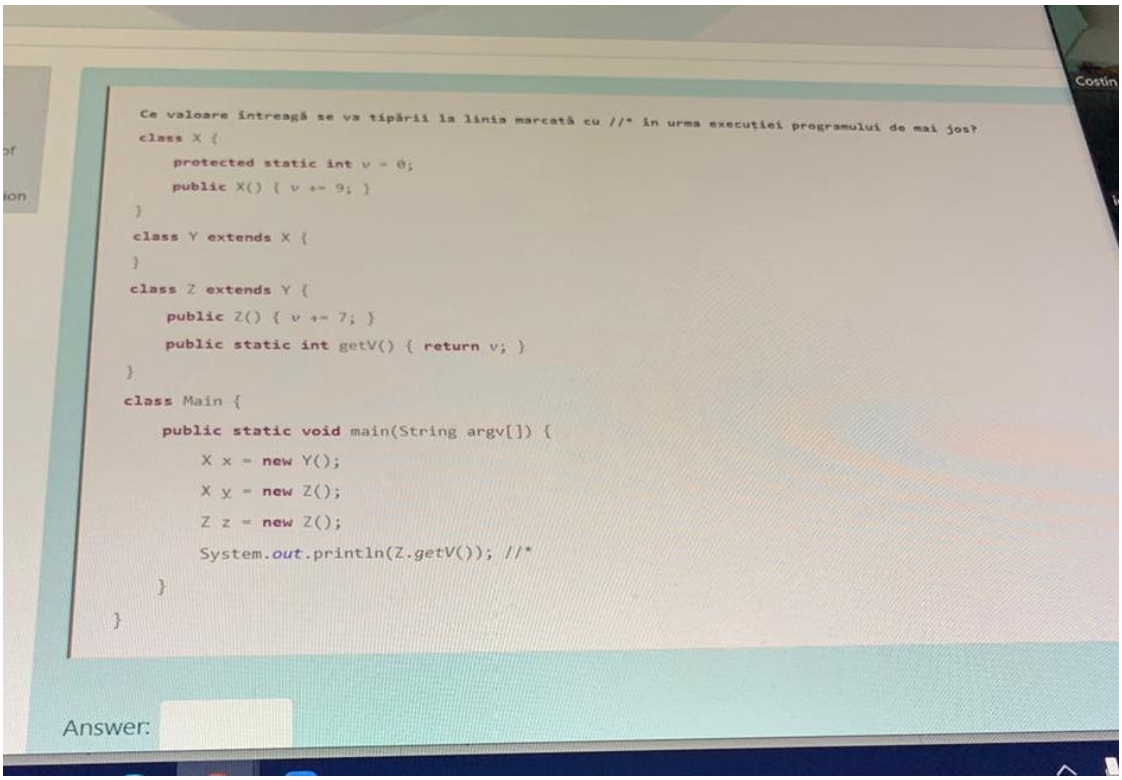
- [ ] `c = b;`
- [ ] `A a = new A();`
- [ ] `c = d;`
- [ ] `A a = d;`
- [ ] `d = c;`


7 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag
Timp rămas 0:09:03
Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor conduce la o eroare de EXECUȚIE a programului când apar la poziția indicată cu /*AICI*/?
class A {
public void m() {}
public static void main(String argv[]) {
A[] a = new A[2];
/*AICI*/
}
}
Selectați unul sau mai multe:
a[0] = new A();
a[0].m();
3°C Cloudy
ENG
13:01
04/02/2022


a[0] = new A();
Timp rămas 0:08:57
a[0].m();
a[2] = new A();
A[] x = new A[3];
x = a;
x[2] = new A();
a[1] = new A();
3°C Cloudy
ENG
13:02
04/02/2022


Unmute
Stop Video
Participants
40
Chat
New Share
Pause Share
Annotate
Remote Control
More
You are screen sharing
Stop Share
Timp rămas
Considerând următorul cod Java, care din afirmațiile enumerate mai jos sunt ADEVĂRATE?
//A.java
package p1;
public class A {
protected int x;
}
//B.java
package p1;
public class B extends A {}
//C.java
package p1;
public class C {}
class D {}
//E.java
package p1.p2;
import p1.A;
import p1.B;
public class E extends B {
43°F Mostly s...


Unmute
Stop Video
Participants
Chat
New Share
Pause Share
Annotate
Remote Control
You are screen sharing
Stop Share
class D {}
Timp rămas 0:02:04
//E.java
package p1.p2;
import p1.A;
import p1.B;
public class E extends B {
E() {}
public void m(A a) {
//1
}
public void n(C c) {} //2
}
Selectați unul sau mai multe:
Clasa E poate fi instanțiată în orice pachet
Linia marcată cu //2 conține o eroare de compilare
La linia marcată cu //1 putem accesa câmpul x al obiectului referit de a
Clasele A și B pot fi implementate în aceeași unitate de compilare, fără a fi necesare alte modificări a programului
Putem declara referințe de tip D doar în cadrul pachetului p1


Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?
class X {
protected static int v = 0;
public X() { v += 9; }
}
class Y extends X {
}
class Z extends Y {
public Z() { v += 7; }
public static int getV() { return v; }
}
class Main {
public static void main(String argv[]) {
X x = new Y();
X y = new Z();
Z z = new Z();
System.out.println(Z.getV()); //*
}
}
Answer:

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?

```
class Pair {
    private int v;
    private int t[];
    public Pair(int x, int [] y) { v = x; t = y; }
    public void setVT(int x, int[] y) { v = x; t = y; }
    public void setVIT(int x, int i, int y) { v = x; t[i] = y; }
    public int getV() { return v; }
    public int[] getT() { return t; }
}
class Main {
    public static Pair call(Pair p, Pair q) {
        q.setVT(48, p.getT());
        q = p;
        q.setVIT(87, 1, 41);
        int tz[] = {93, 81};
        return new Pair(38, tz);
    }
    public static void main(String[] args) {
        int ta[] = {39, 45};
        int tb[] = {11, 76};
        Pair a = new Pair(55, ta);
        Pair b = new Pair(35, tb);
        Pair c = b;
        b = call(a, b);
        System.out.println(c.getV()); //*
    }
}
```

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?

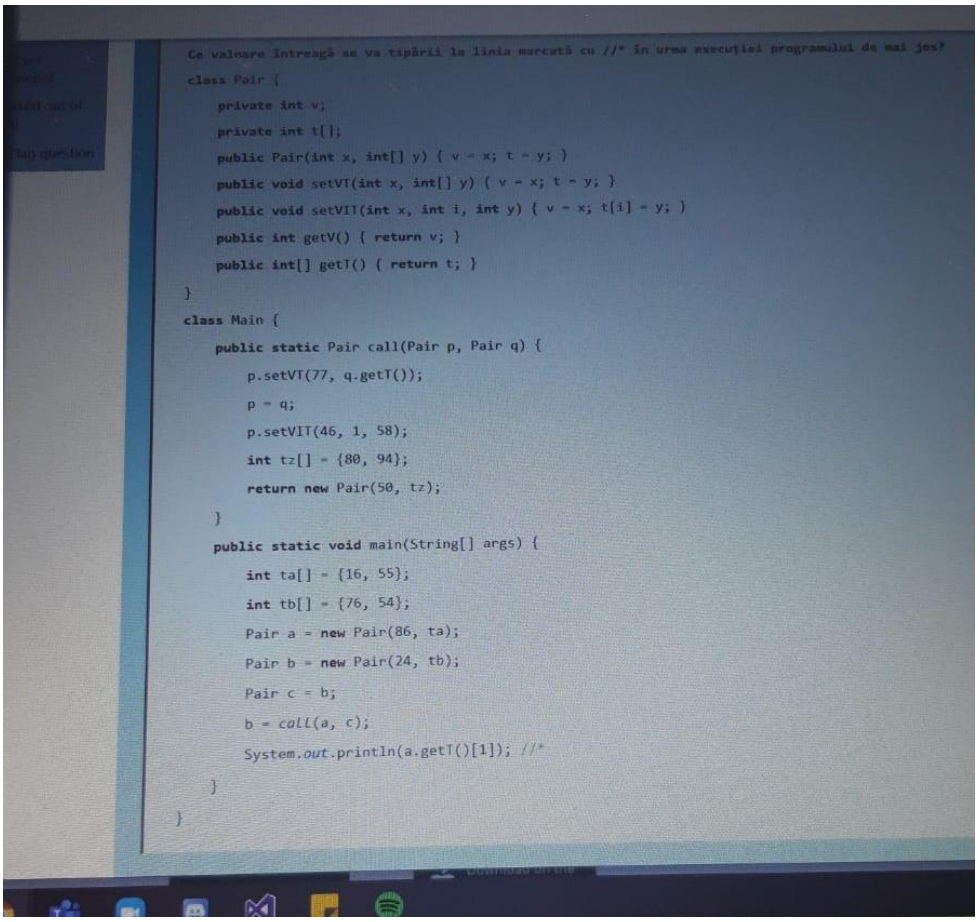
```
abstract class A {
    public int proc(A p) { return 98; }
}
class B extends A {
    public int proc(A p) { return 17;}
}
class C extends A {
    public int proc(C p) { return 65; }
}
public class Main {
    public static void main(String argv[]) {
        C x = new C();
        A y = new B();
        C z = new C();
        System.out.println(y.proc(x) + z.proc(x)); //*
    }
}
```

Answer:

Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?

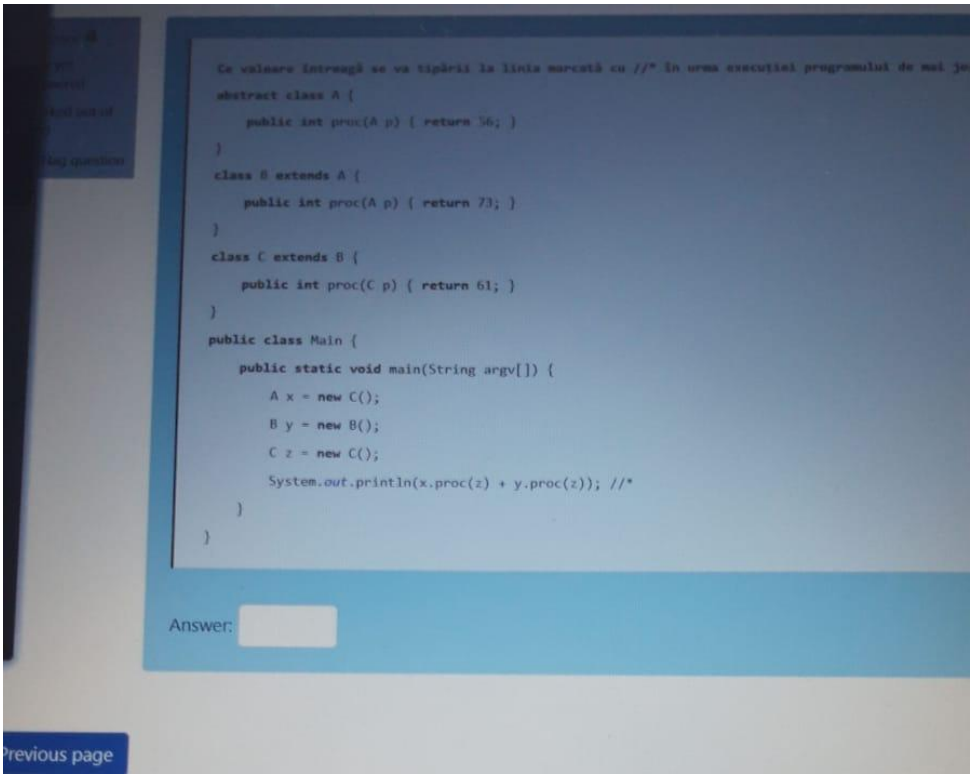
```
class X {
    protected int v = 0;
    public X() { v += 87; }
}
class Y extends X {
    public Y() { v += 94; }
    public int getV() { return v; }
}
class Z extends Y {
    public Z() { v += 46; }
}
class Main {
    public static void main(String argv[]) {
        X x = new Z();
        Y y = new Y();
        Z z = new Z();
        System.out.println(y.getV()); //*
    }
}
```

Answer:



Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?

```
class Pair {
    private int v;
    private int t[];
    public Pair(int x, int[] y) { v = x; t = y; }
    public void setVT(int x, int[] y) { v = x; t = y; }
    public void setVIT(int x, int i, int y) { v = x; t[i] = y; }
    public int getV() { return v; }
    public int[] getT() { return t; }
}
class Main {
    public static Pair call(Pair p, Pair q) {
        p.setVT(77, q.getT());
        p = q;
        p.setVIT(46, 1, 58);
        int tz[] = {80, 94};
        return new Pair(50, tz);
    }
    public static void main(String[] args) {
        int ta[] = {16, 55};
        int tb[] = {76, 54};
        Pair a = new Pair(86, ta);
        Pair b = new Pair(24, tb);
        Pair c = b;
        b = call(a, c);
        System.out.println(a.getT()[1]); //*
    }
}
```

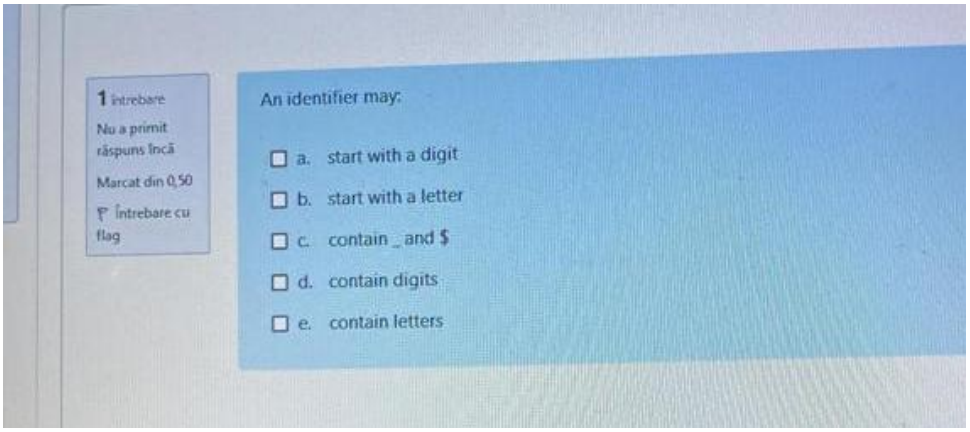


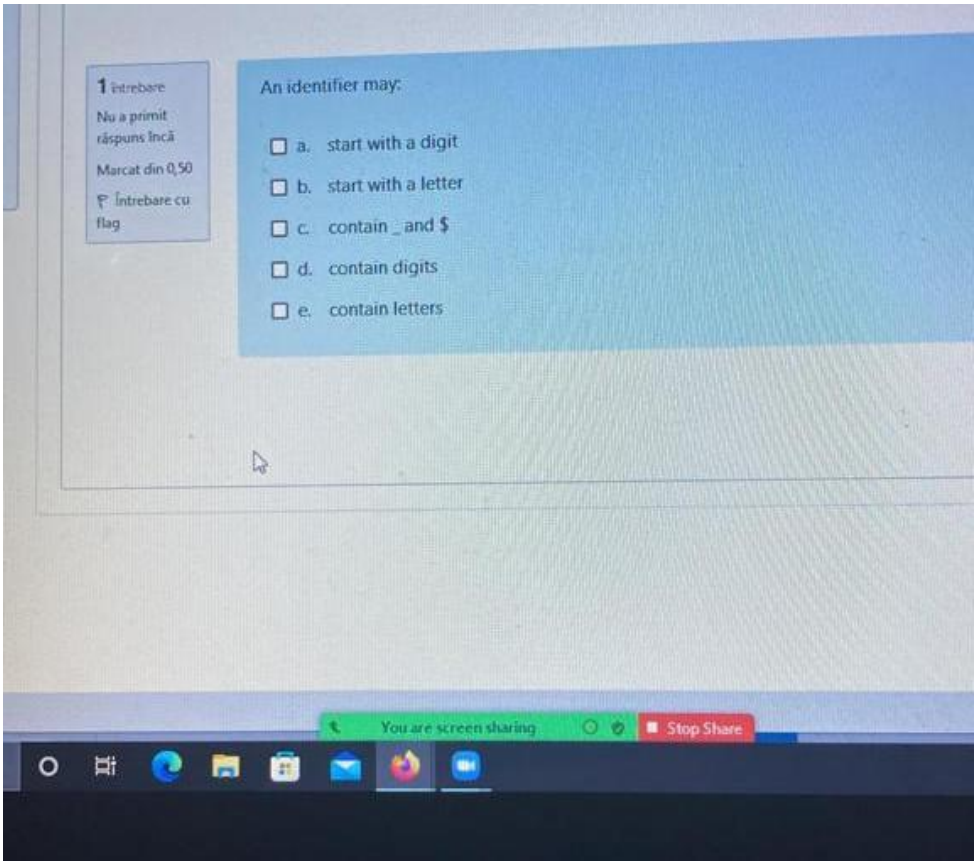
Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jo

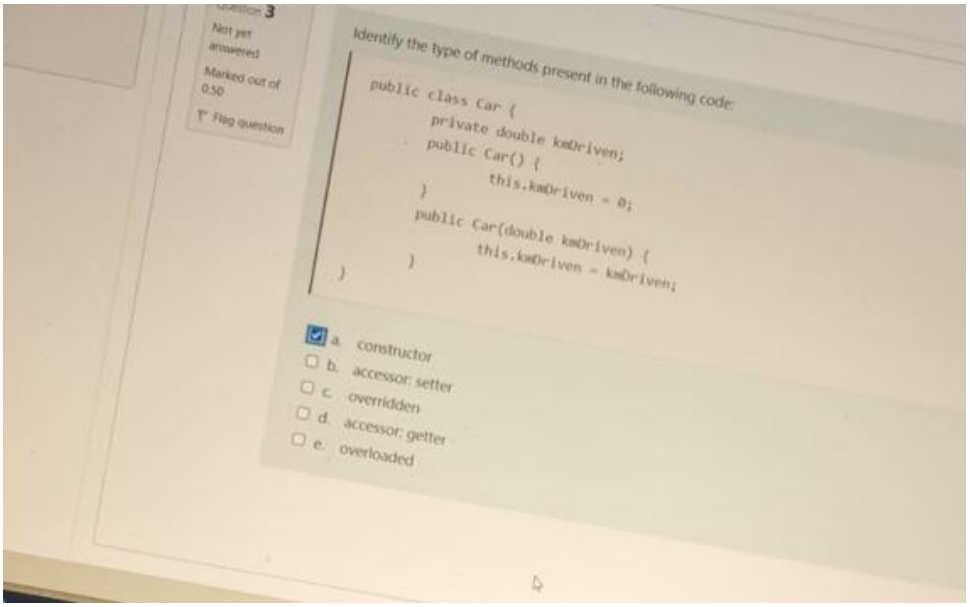
```
abstract class A {
    public int proc(A p) { return 56; }
}
class B extends A {
    public int proc(A p) { return 73; }
}
class C extends B {
    public int proc(C p) { return 61; }
}
public class Main {
    public static void main(String argv[]) {
        A x = new C();
        B y = new B();
        C z = new C();
        System.out.println(x.proc(z) + y.proc(z)); //*
    }
}
```

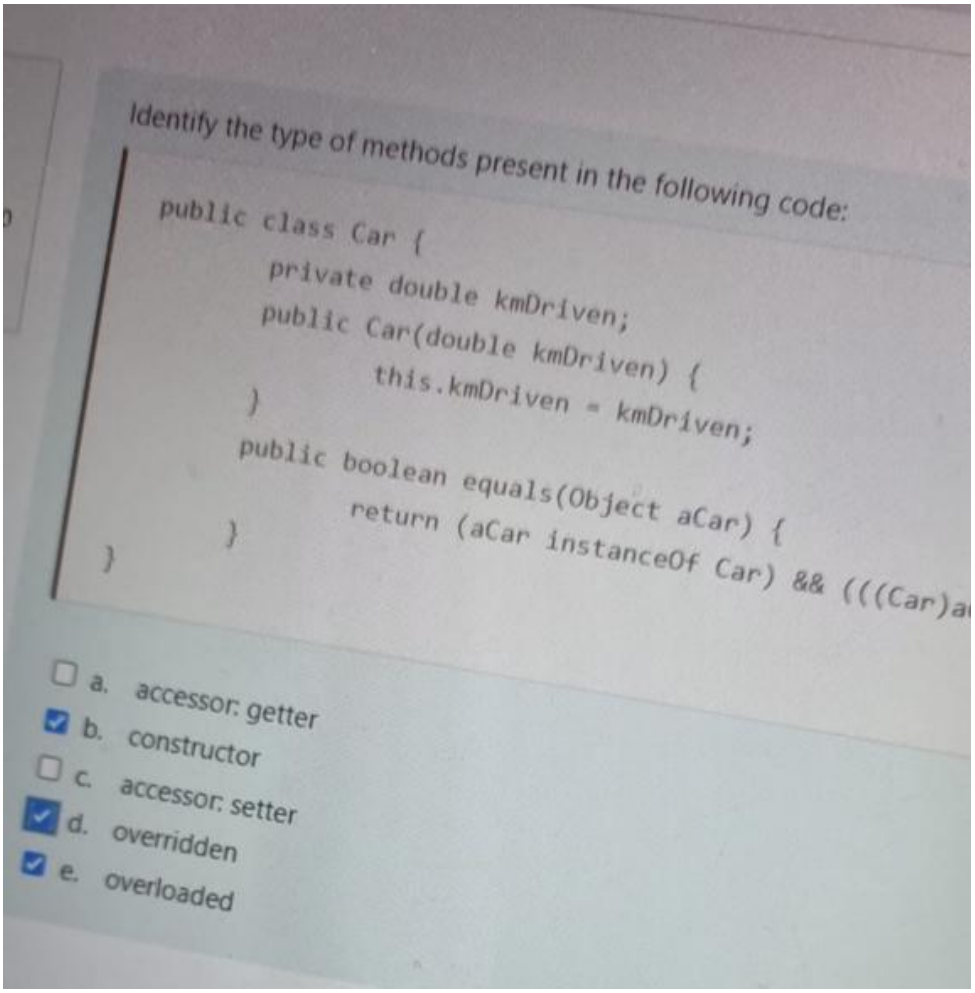
Answer:

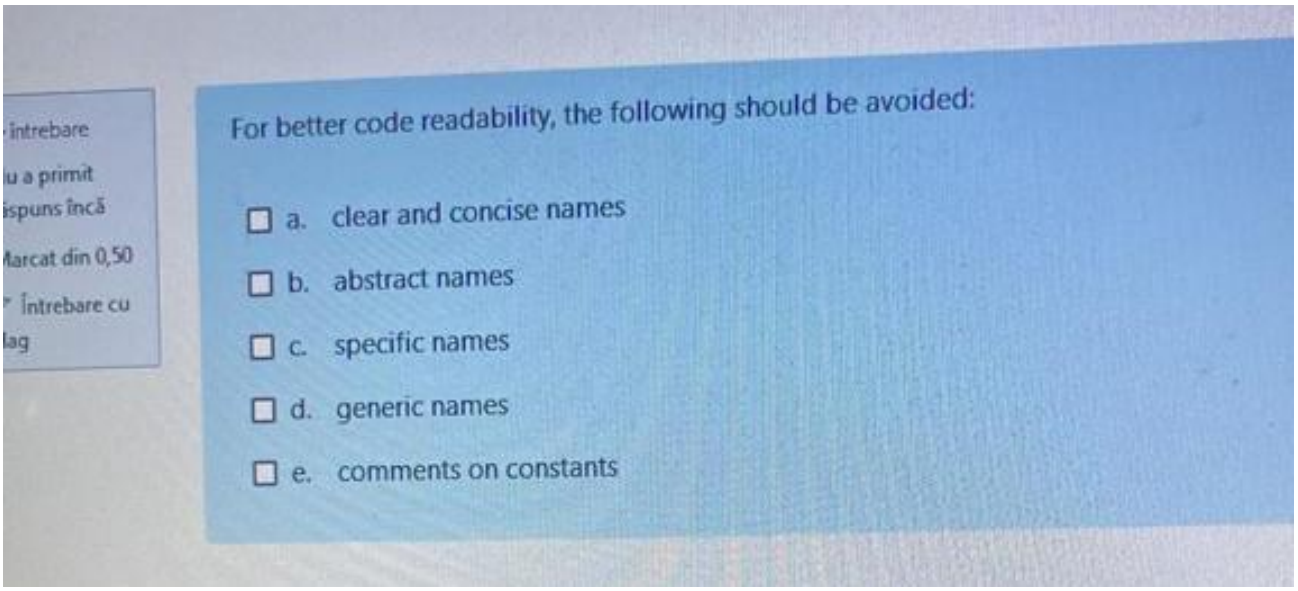
Previous page

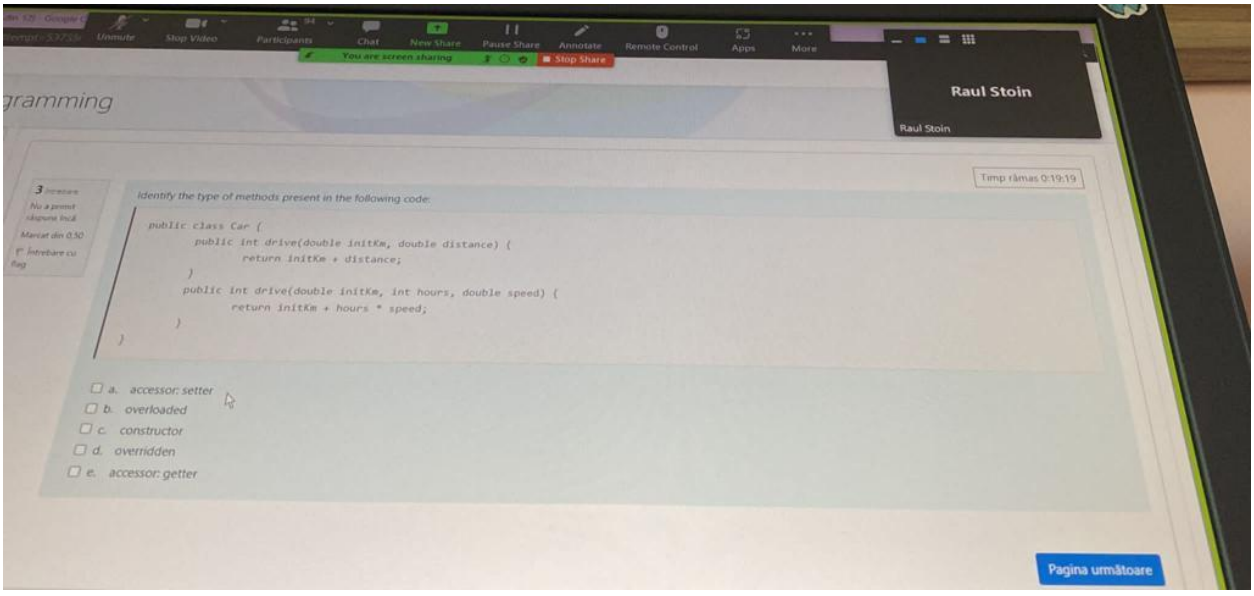

1 întrebare
Nu a primit răspuns încă
Marcat din 0,50
Întrebare cu flag
An identifier may:
a. start with a digit
b. start with a letter
c. contain _ and $
d. contain digits
e. contain letters

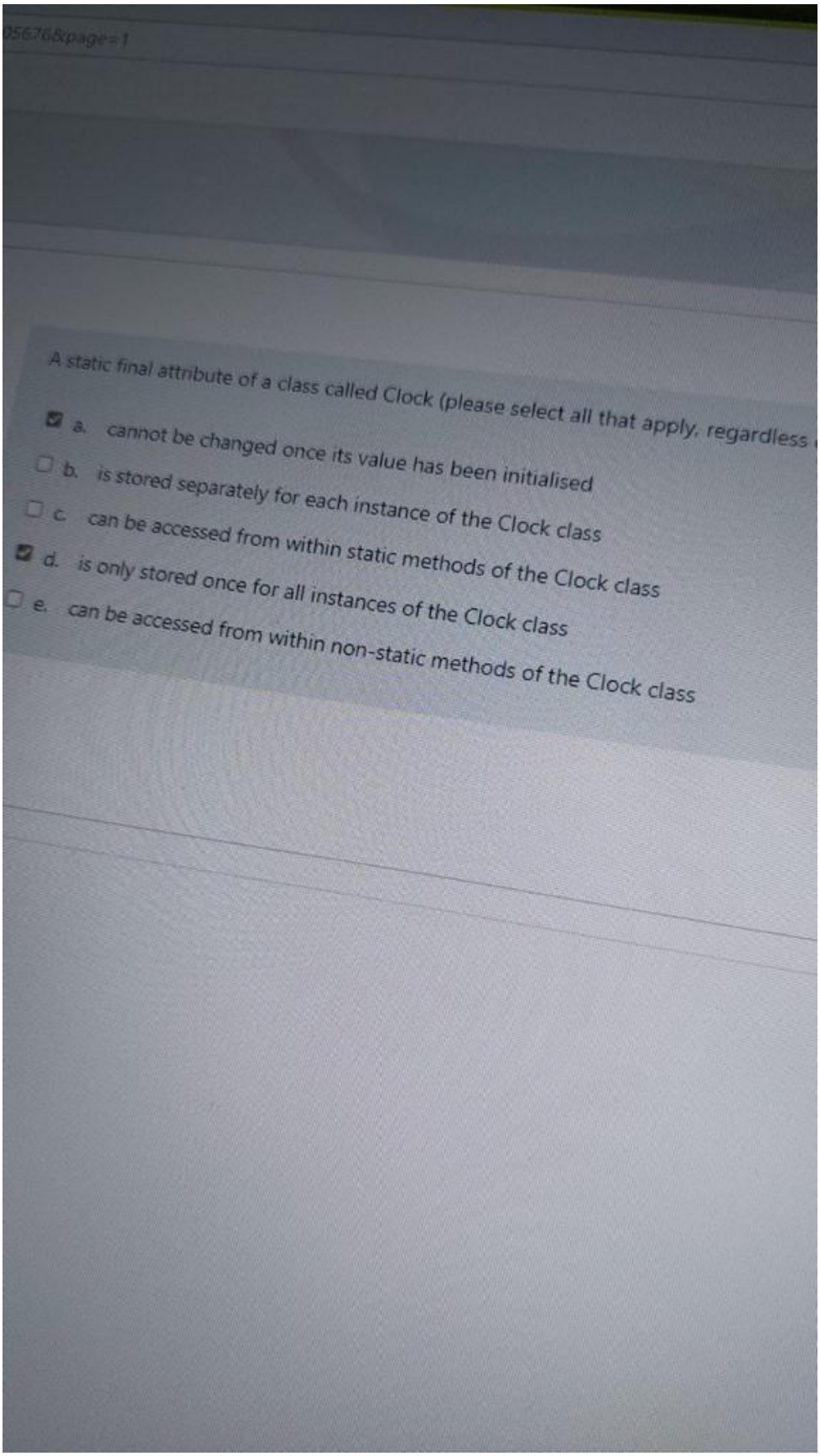

1 întrebare
Nu a primit răspuns încă
Marcat din 0,50
Întrebare cu flag
An identifier may:
a. start with a digit
b. start with a letter
c. contain _ and $
d. contain digits
e. contain letters
You are screen sharing
Stop Share

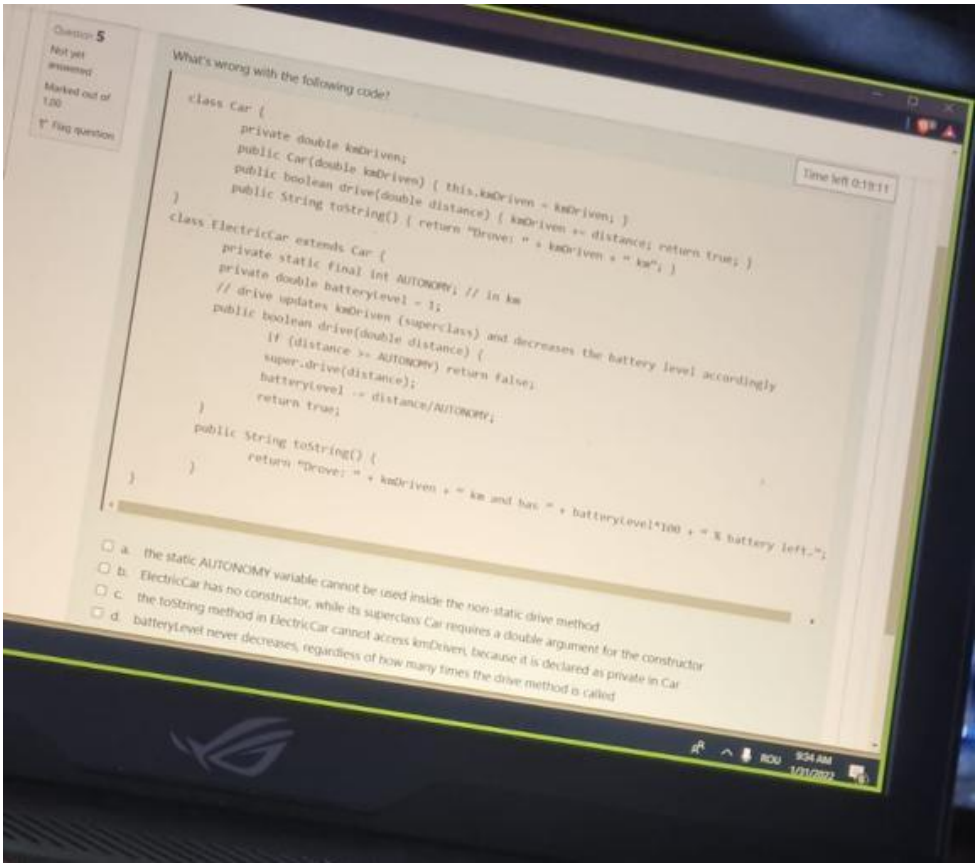

Question 3
Not yet answered
Marked out of 0.50
Flag question
Identify the type of methods present in the following code:
public class Car {
private double kmDriven;
public Car() {
this.kmDriven = 0;
}
public Car(double kmDriven) {
this.kmDriven = kmDriven;
}
}
a. constructor
b. accessor: setter
c. overridden
d. accessor: getter
e. overloaded


Identify the type of methods present in the following code:
public class Car {
private double kmDriven;
public Car(double kmDriven) {
this.kmDriven = kmDriven;
}
public boolean equals(Object aCar) {
return (aCar instanceOf Car) && (((Car)a
}
}
a. accessor: getter
b. constructor
c. accessor: setter
d. overridden
e. overloaded


întrebare
u a primit
ispuns încă
Marcat din 0,50
Întrebare cu
lag
For better code readability, the following should be avoided:
a. clear and concise names
b. abstract names
c. specific names
d. generic names
e. comments on constants


Unmute
Stop Video
Participants
Chat
New Share
Pause Share
Annotate
Remote Control
Apps
More
You are screen sharing
Stop Share
Raul Stoin
Raul Stoin
gramming
Timp rămas 0:19:19
Identify the type of methods present in the following code:
public class Car {
public int drive(double initKm, double distance) {
return initKm + distance;
}
public int drive(double initKm, int hours, double speed) {
return initKm + hours * speed;
}
}
a. accessor: setter
b. overloaded
c. constructor
d. overridden
e. accessor: getter
Pagina următoare


05676&page=1
A static final attribute of a class called Clock (please select all that apply, regardless
a. cannot be changed once its value has been initialised
b. is stored separately for each instance of the Clock class
c. can be accessed from within static methods of the Clock class
d. is only stored once for all instances of the Clock class
e. can be accessed from within non-static methods of the Clock class


Question 5
Not yet answered
Marked out of 1.00
Flag question
What's wrong with the following code?
Time left 0:19:11
class Car {
private double kmDriven;
public Car(double kmDriven) { this.kmDriven = kmDriven; }
public boolean drive(double distance) { kmDriven += distance; return true; }
public String toString() { return "Drove: " + kmDriven + " km"; }
}
class ElectricCar extends Car {
private static final int AUTONOMY; // in km
private double batteryLevel = 1;
// drive updates kmDriven (superclass) and decreases the battery level accordingly
public boolean drive(double distance) {
if (distance >= AUTONOMY) return false;
super.drive(distance);
batteryLevel -= distance/AUTONOMY;
return true;
}
public String toString() {
return "Drove: " + kmDriven + " km and has " + batteryLevel*100 + "% battery left.";
}
}
a. the static AUTONOMY variable cannot be used inside the non-static drive method
b. ElectricCar has no constructor, while its superclass Car requires a double argument for the constructor
c. the toString method in ElectricCar cannot access kmDriven, because it is declared as private in Car
d. batteryLevel never decreases, regardless of how many times the drive method is called
ROU
9:34 AM
3/31/2021

's wrong with the following code?

```
lass Car {
    private double kmDriven;
    public Car(double kmDriven) { this.kmDriven = kmDriven; }
    public boolean drive(double distance) { kmDriven += distance; return true; }
    public String toString() { return "Drove: " + kmDriven + " km"; }

lass ElectricCar extends Car {
    private static final int AUTONOMY; // in km
    private double batteryLevel = 1;
    // drive updates kmDriven (superclass) and decreases the battery level accordingly
    public boolean drive(double distance) {
        if (distance >= AUTONOMY) return false;
        super.drive(distance);
        batteryLevel -= distance/AUTONOMY;
        return true;
    }
    public String toString() {
        return "Drove: " + kmDriven + " km and has " + batteryLevel*100 + " % battery left.";
    }
```

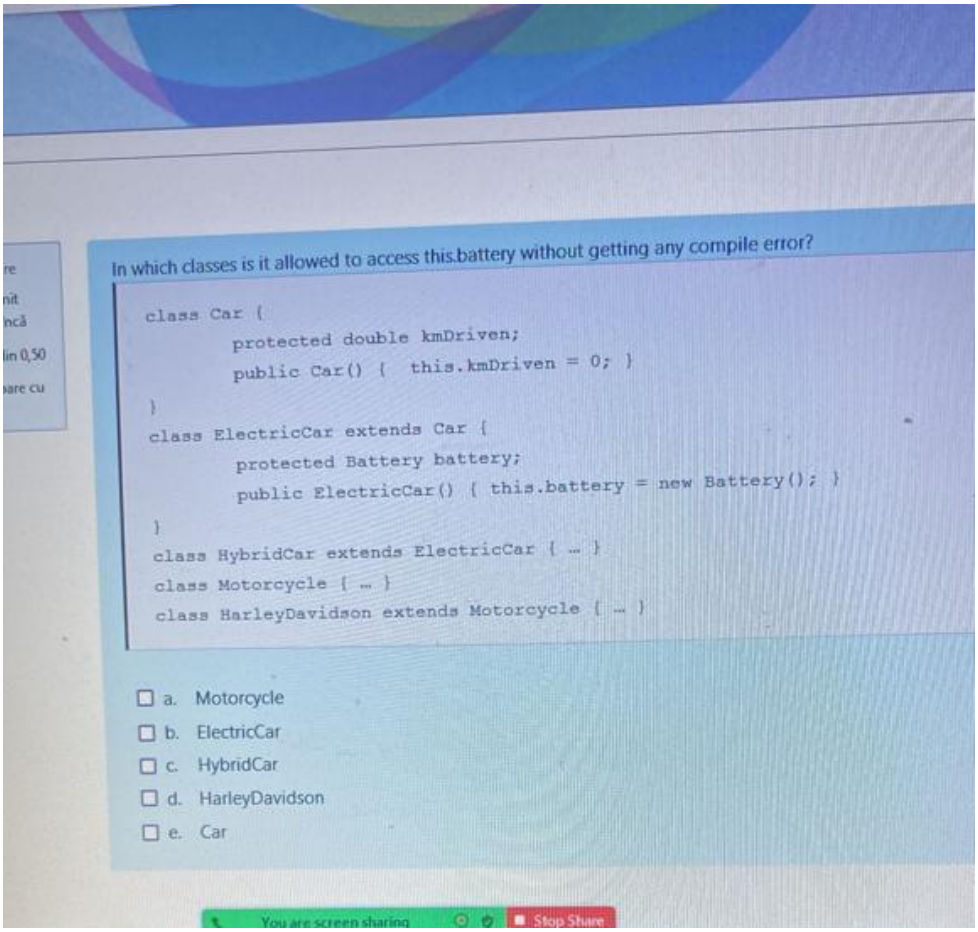
. the static AUTONOMY variable cannot be used inside the non-static drive method

. ElectricCar has no constructor, while its superclass Car requires a double argument for the constructor

. the final AUTONOMY variable needs to be initialised

. the toString method in ElectricCar cannot access kmDriven, because it is declared as private in Car

. batteryLevel never decreases, regardless of how many times the drive method is called

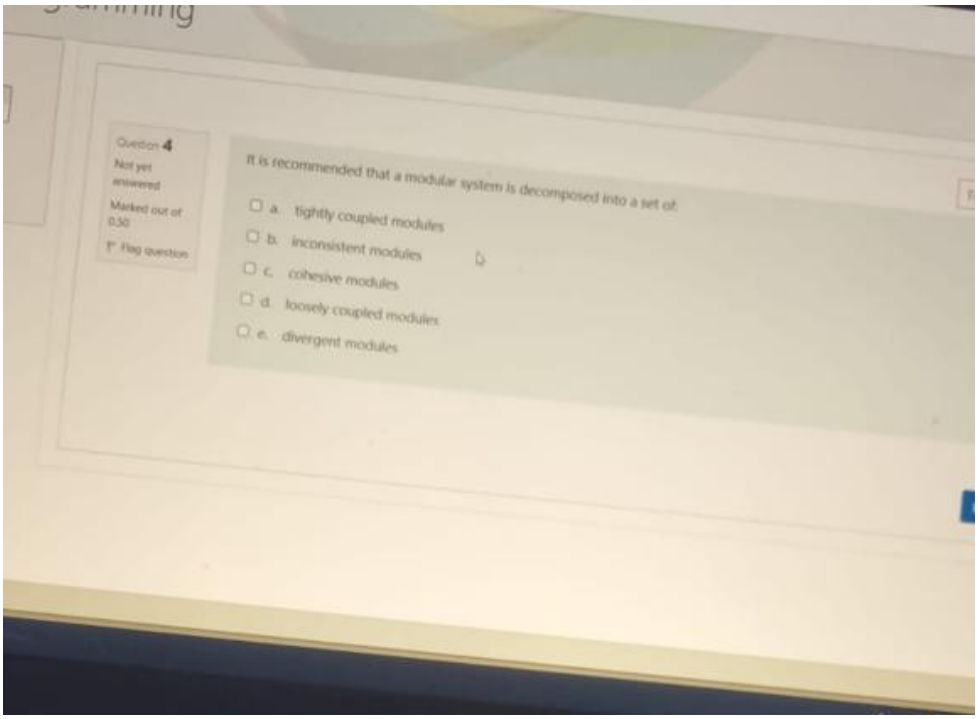


In which classes is it allowed to access this.battery without getting any compile error?

```
class Car {
        protected double kmDriven;
        public Car() {  this.kmDriven = 0; }
}
class ElectricCar extends Car {
        protected Battery battery;
        public ElectricCar() { this.battery = new Battery(); }
}
class HybridCar extends ElectricCar { … }
class Motorcycle { … }
class HarleyDavidson extends Motorcycle { … }
```

- a. Motorcycle
- b. ElectricCar
- c. HybridCar
- d. HarleyDavidson
- e. Car

You are screen sharing  Stop Share



Question 4
Not yet answered
Marked out of 0.50
Flag question

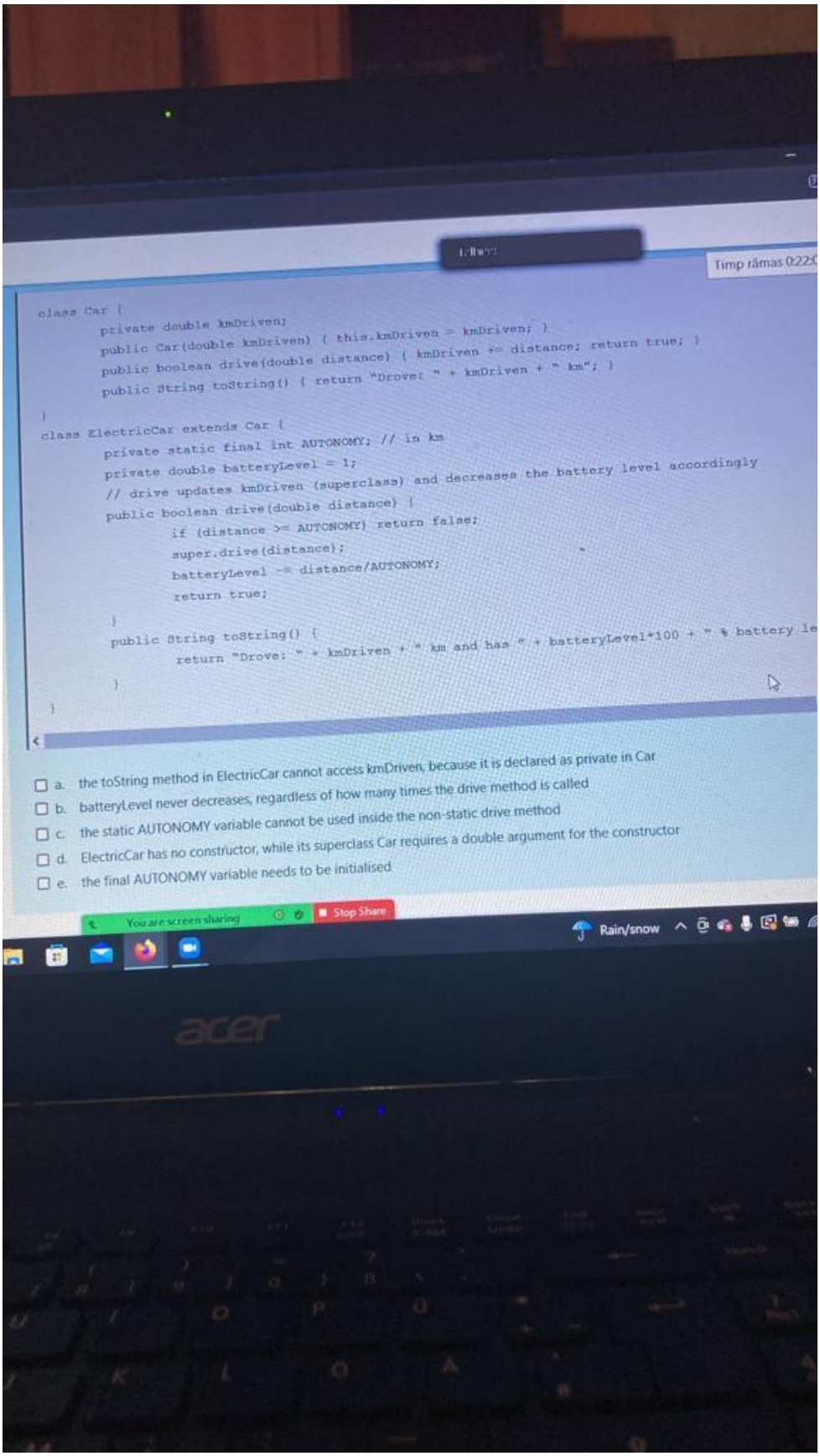
It is recommended that a modular system is decomposed into a set of

- a. tightly coupled modules
- b. inconsistent modules
- c. cohesive modules
- d. loosely coupled modules
- e. divergent modules

Identify the type of methods present in the following code:

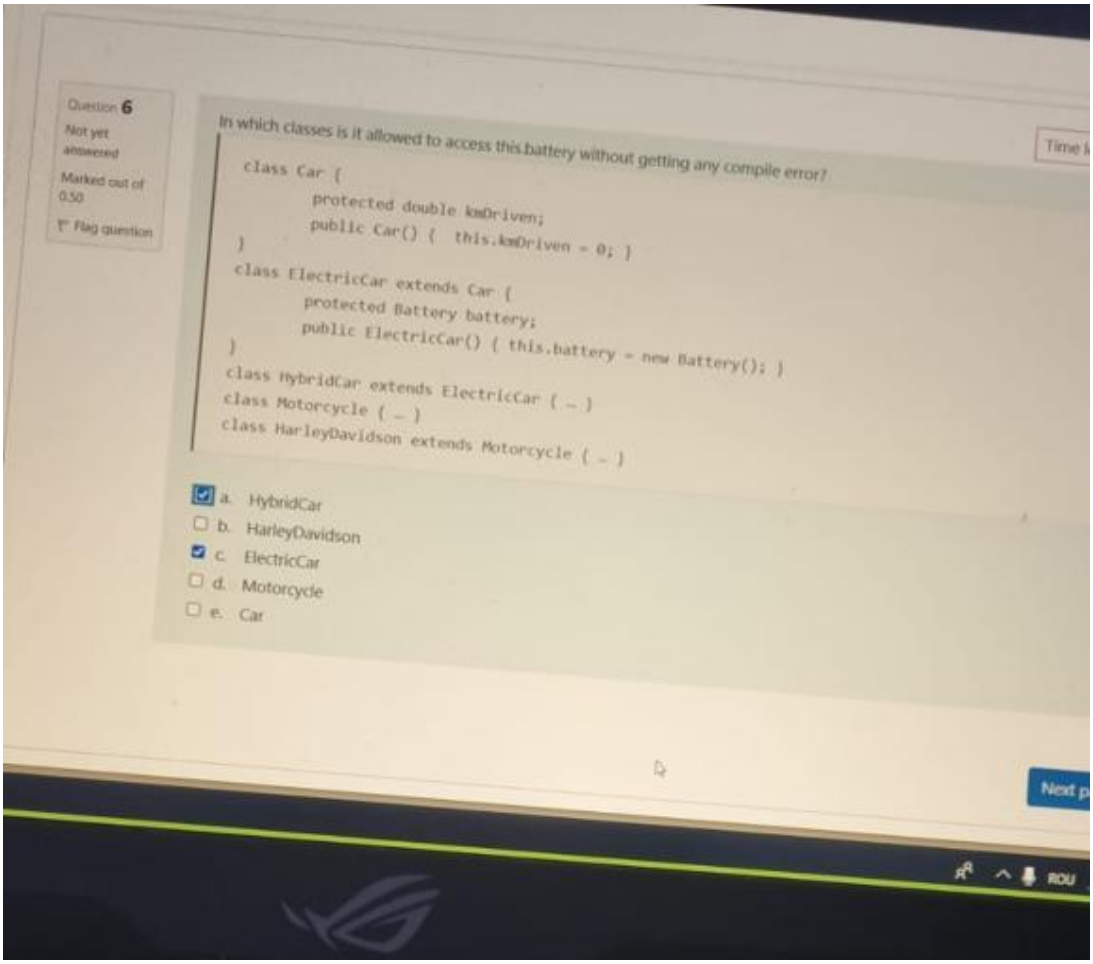
```
public class Car {
        private double kmDriven;
        public Car() {
                this.kmDriven = 0;
        }
        public Car(double kmDriven) {
                this.kmDriven = kmDriven;
        }
}
```

- [x] a. constructor
- [ ] b. overridden
- [x] c. overloaded
- [ ] d. accessor: getter
- [ ] e. accessor: setter



```
class Car {
        private double kmDriven;
        public Car(double kmDriven) { this.kmDriven = kmDriven; }
        public boolean drive(double distance) { kmDriven += distance; return true; }
        public String toString() { return "Drove: " + kmDriven + " km"; }
}
class ElectricCar extends Car {
        private static final int AUTONOMY; // in km
        private double batteryLevel = 1;
        // drive updates kmDriven (superclass) and decreases the battery level accordingly
        public boolean drive(double distance) {
                if (distance >= AUTONOMY) return false;
                super.drive(distance);
                batteryLevel -= distance/AUTONOMY;
                return true;
        }
        public String toString() {
                return "Drove: " + kmDriven + " km and has " + batteryLevel*100 + " % battery le
        }
}
```

- [ ] a. the toString method in ElectricCar cannot access kmDriven, because it is declared as private in Car
- [ ] b. batteryLevel never decreases, regardless of how many times the drive method is called
- [ ] c. the static AUTONOMY variable cannot be used inside the non-static drive method
- [ ] d. ElectricCar has no constructor, while its superclass Car requires a double argument for the constructor
- [ ] e. the final AUTONOMY variable needs to be initialised



Question 6

Not yet answered

Marked out of 0.50

Flag question

In which classes is it allowed to access this.battery without getting any compile error?

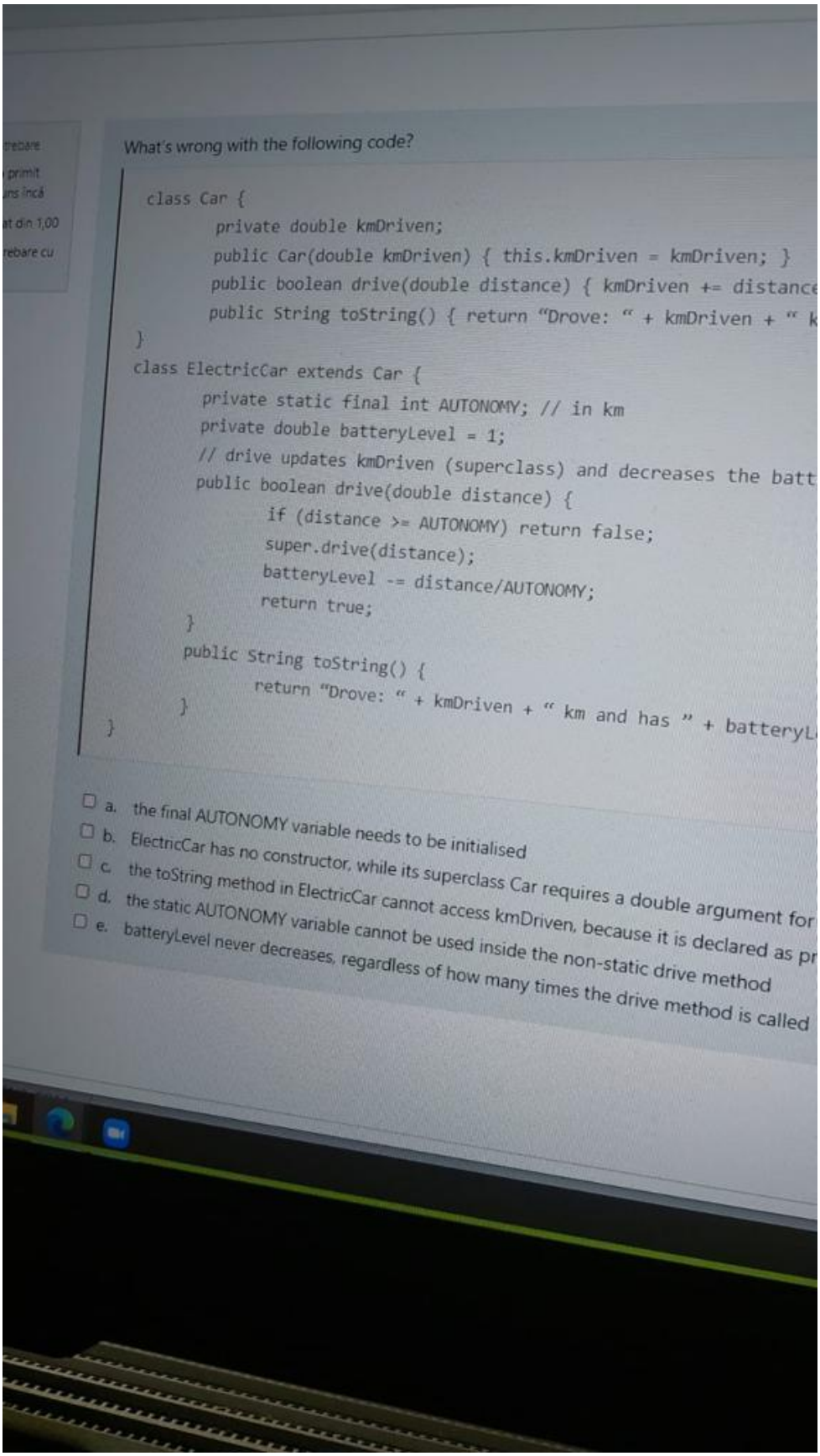
```
class Car {
        protected double kmDriven;
        public Car() {  this.kmDriven = 0; }
}
class ElectricCar extends Car {
        protected Battery battery;
        public ElectricCar() { this.battery = new Battery(); }
}
class HybridCar extends ElectricCar { … }
class Motorcycle { … }
class HarleyDavidson extends Motorcycle { … }
```

- [x] a. HybridCar
- [ ] b. HarleyDavidson
- [x] c. ElectricCar
- [ ] d. Motorcycle
- [ ] e. Car

Time

Next p

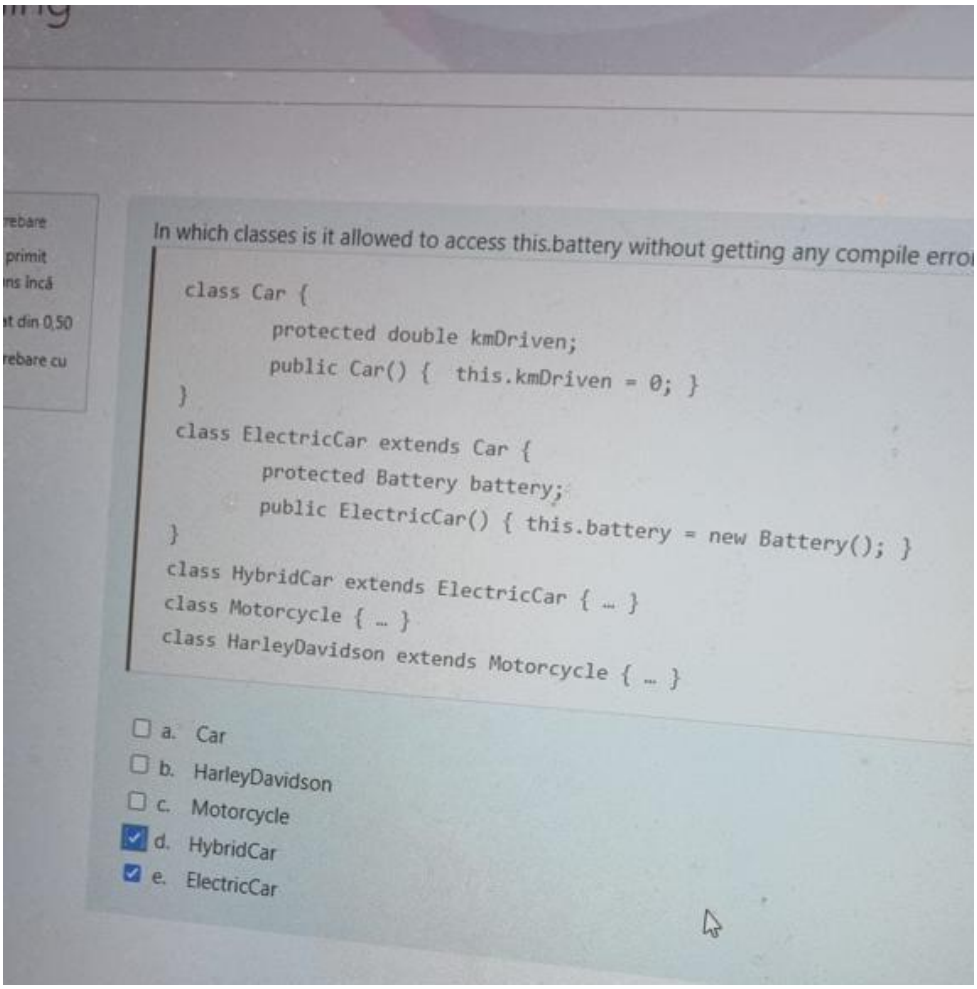
ROU



What's wrong with the following code?

```
class Car {
        private double kmDriven;
        public Car(double kmDriven) { this.kmDriven = kmDriven; }
        public boolean drive(double distance) { kmDriven += distance
        public String toString() { return "Drove: " + kmDriven + " k
}
class ElectricCar extends Car {
        private static final int AUTONOMY; // in km
        private double batteryLevel = 1;
        // drive updates kmDriven (superclass) and decreases the batt
        public boolean drive(double distance) {
                if (distance >= AUTONOMY) return false;
                super.drive(distance);
                batteryLevel -= distance/AUTONOMY;
                return true;
        }
        public String toString() {
                return "Drove: " + kmDriven + " km and has " + batteryL
        }
}
```

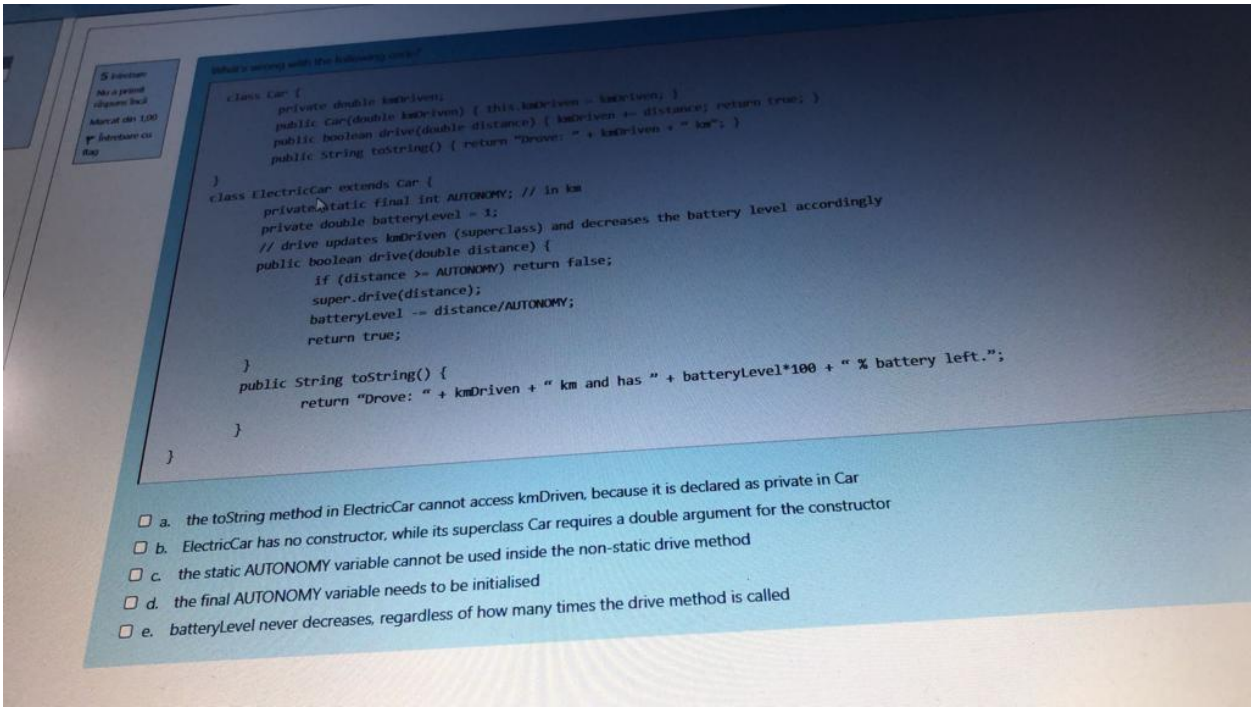
- ☐ a. the final AUTONOMY variable needs to be initialised
- ☐ b. ElectricCar has no constructor, while its superclass Car requires a double argument for
- ☐ c. the toString method in ElectricCar cannot access kmDriven, because it is declared as pr
- ☐ d. the static AUTONOMY variable cannot be used inside the non-static drive method
- ☐ e. batteryLevel never decreases, regardless of how many times the drive method is called



In which classes is it allowed to access this.battery without getting any compile erro

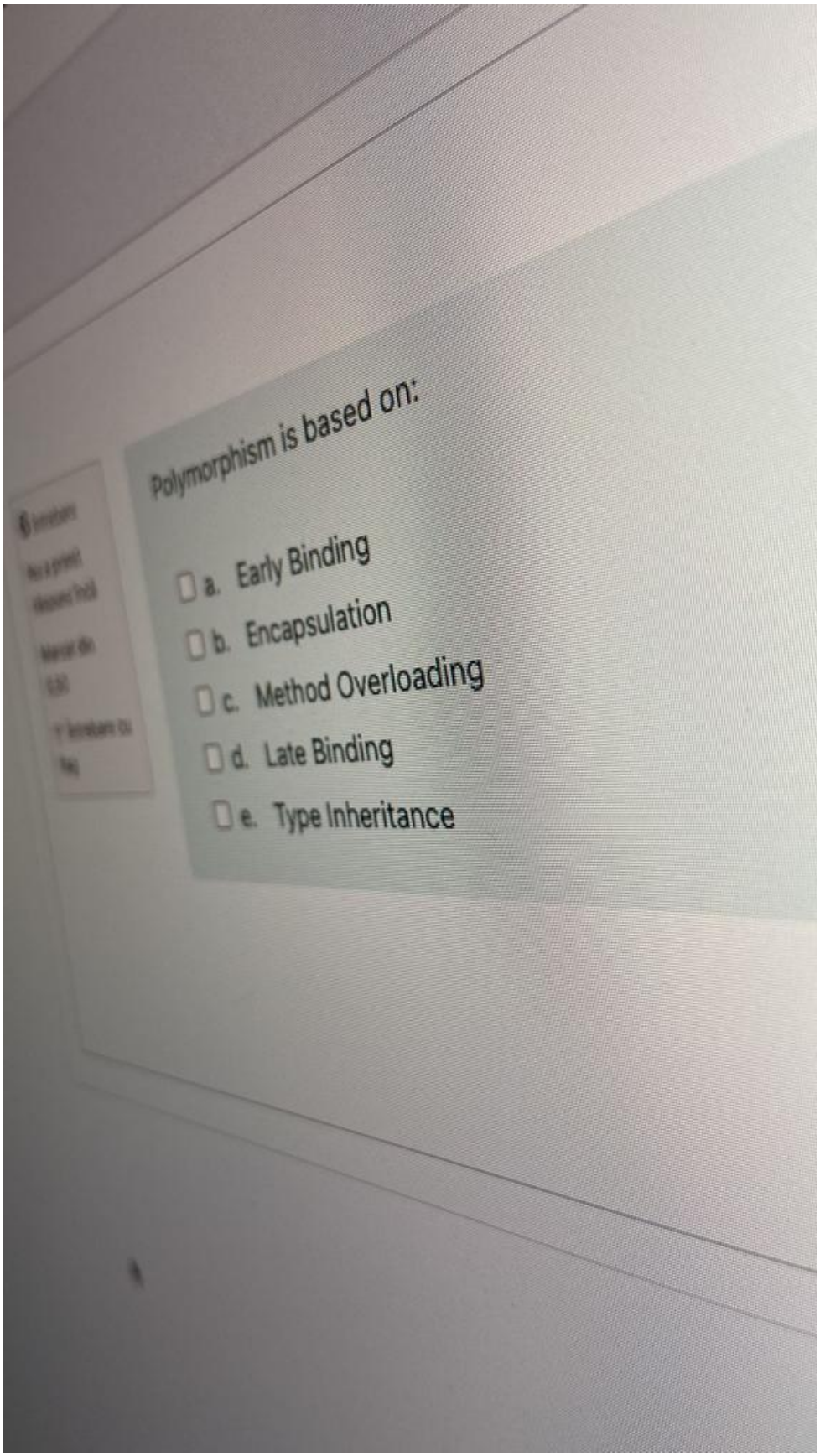
```
class Car {
        protected double kmDriven;
        public Car() {  this.kmDriven = 0; }
}
class ElectricCar extends Car {
        protected Battery battery;
        public ElectricCar() { this.battery = new Battery(); }
}
class HybridCar extends ElectricCar { … }
class Motorcycle { … }
class HarleyDavidson extends Motorcycle { … }
```

- [ ] a. Car
- [ ] b. HarleyDavidson
- [ ] c. Motorcycle
- [x] d. HybridCar
- [x] e. ElectricCar



```
class Car {
        private double kmDriven;
        public Car(double kmDriven) { this.kmDriven = kmDriven; }
        public boolean drive(double distance) { kmDriven += distance; return true; }
        public String toString() { return "Drove: " + kmDriven + " km"; }
}
class ElectricCar extends Car {
        private static final int AUTONOMY; // in km
        private double batteryLevel = 1;
        // drive updates kmDriven (superclass) and decreases the battery level accordingly
        public boolean drive(double distance) {
                if (distance >= AUTONOMY) return false;
                super.drive(distance);
                batteryLevel -= distance/AUTONOMY;
                return true;
        }
        public String toString() {
                return "Drove: " + kmDriven + " km and has " + batteryLevel*100 + " % battery left.";
        }
}
```

- [ ] a. the toString method in ElectricCar cannot access kmDriven, because it is declared as private in Car
- [ ] b. ElectricCar has no constructor, while its superclass Car requires a double argument for the constructor
- [ ] c. the static AUTONOMY variable cannot be used inside the non-static drive method
- [ ] d. the final AUTONOMY variable needs to be initialised
- [ ] e. batteryLevel never decreases, regardless of how many times the drive method is called


Polymorphism is based on:
a. Early Binding
b. Encapsulation
c. Method Overloading
d. Late Binding
e. Type Inheritance

It is recommended that a modular system is decomposed into a set o

- [ ] a. loosely coupled modules
- [ ] b. divergent modules
- [x] c. cohesive modules
- [ ] d. tightly coupled modules
- [ ] e. inconsistent modules

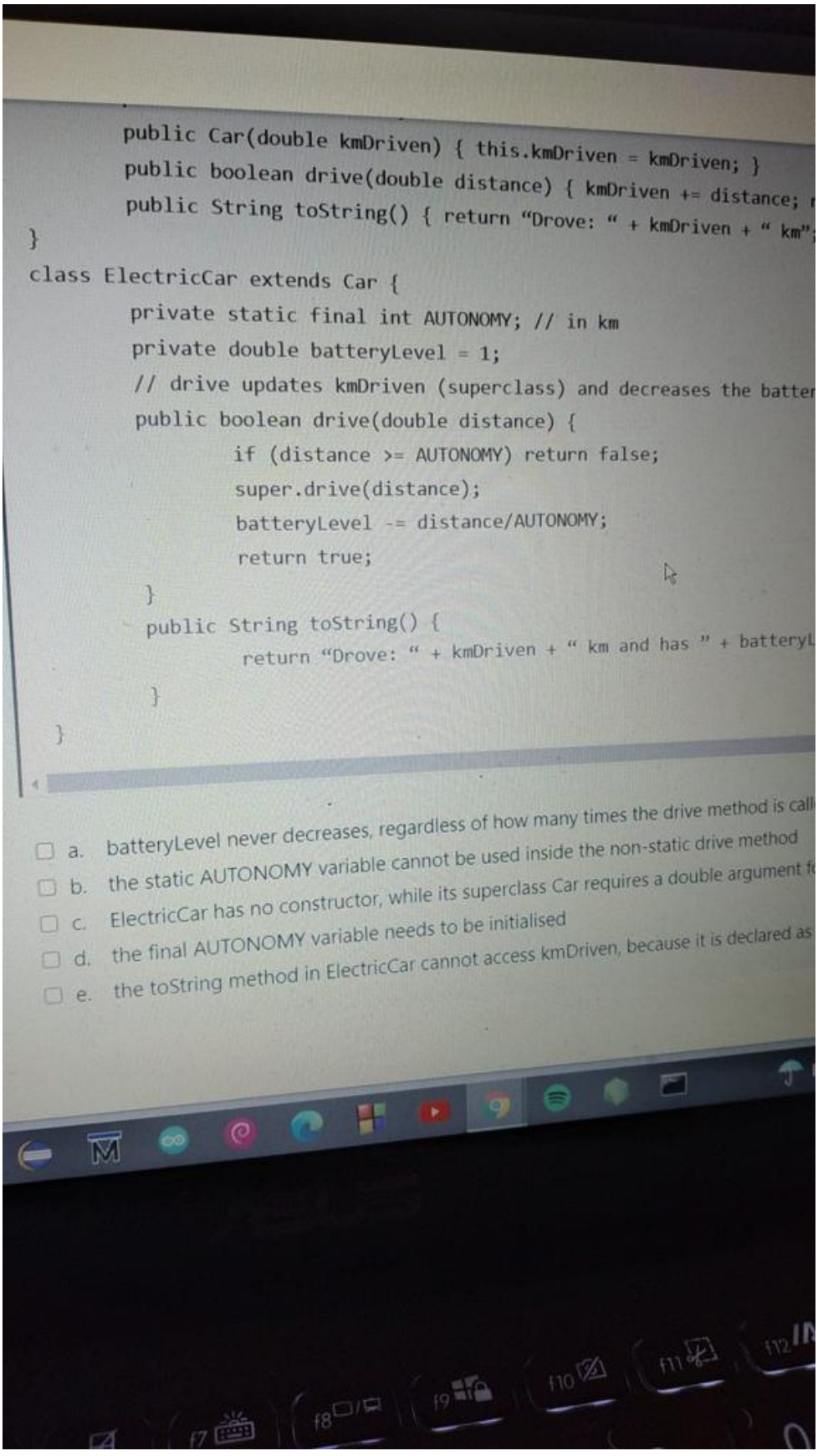
What will be displayed when running the following code?

```
class SuperClass {
        public SuperClass() {
                System.out.println("Constructor from SuperClass");
                methodCall();
        }
        public void methodCall() { System.out.println("Method from SuperClass"); }
}
class SubClass extends SuperClass {
        public SubClass() { System.out.println("Constructor from SubClass"); }
        public void methodCall() { System.out.println("Method from SubClass"); }
}
public class Client {
        public static void main(String[] args) {
                SubClass s = new SubClass();
        }
}
```

- [ ] a. Constructor from SuperClass
  Constructor from SubClass
  Method from SuperClass
- [ ] b. Constructor from SuperClass
  Constructor from SubClass
  Method from SubClass
- [ ] c. Constructor from SubClass
  Constructor from SuperClass
  Method from SuperClass
- [ ] d. Constructor from SuperClass

You are screen sharing  Stop Share

Rain/snow



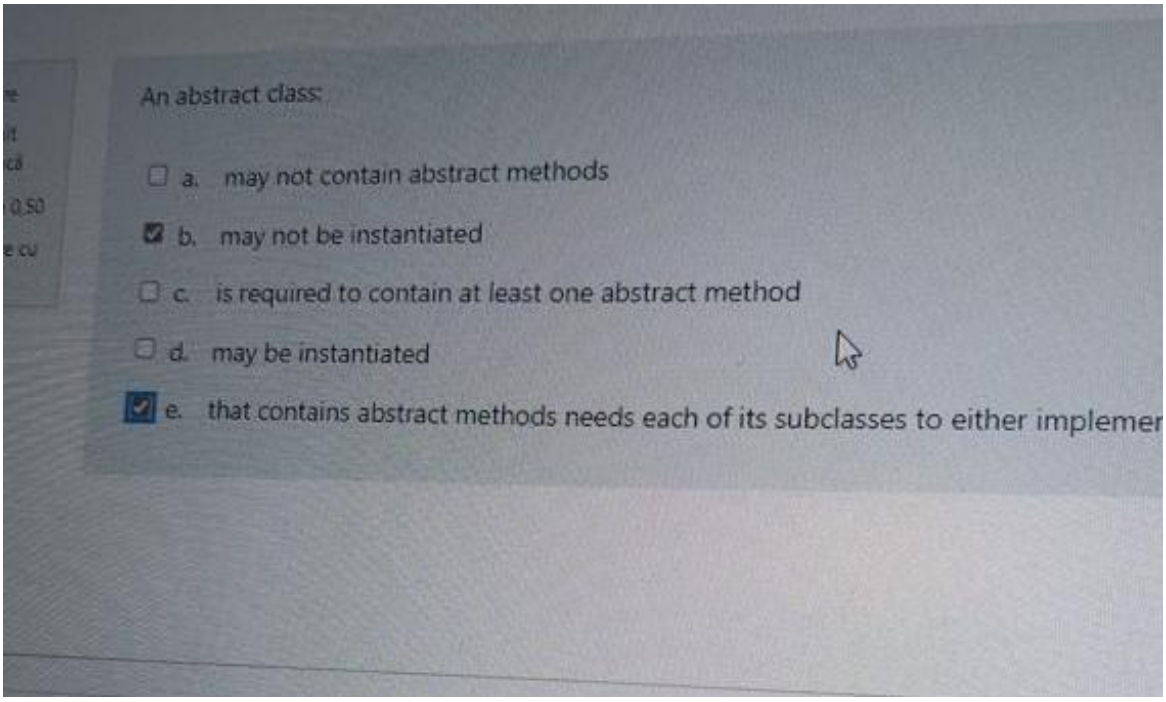
```
        public Car(double kmDriven) { this.kmDriven = kmDriven; }
        public boolean drive(double distance) { kmDriven += distance; r
        public String toString() { return "Drove: " + kmDriven + " km";
}
class ElectricCar extends Car {
        private static final int AUTONOMY; // in km
        private double batteryLevel = 1;
        // drive updates kmDriven (superclass) and decreases the batter
        public boolean drive(double distance) {
                if (distance >= AUTONOMY) return false;
                super.drive(distance);
                batteryLevel -= distance/AUTONOMY;
                return true;
        }
        public String toString() {
                return "Drove: " + kmDriven + " km and has " + batteryL
        }
}
```

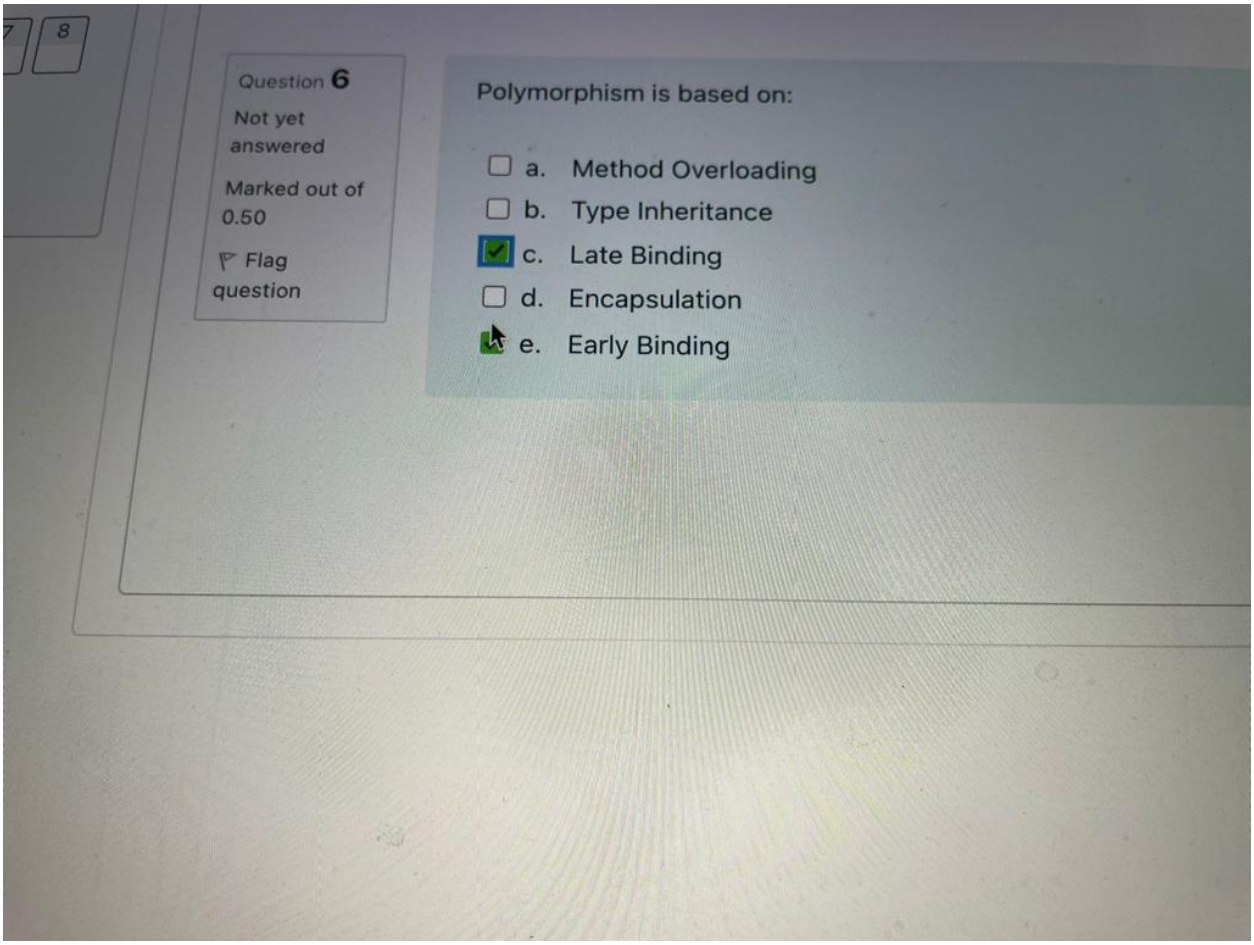
- ☐ a. batteryLevel never decreases, regardless of how many times the drive method is call
- ☐ b. the static AUTONOMY variable cannot be used inside the non-static drive method
- ☐ c. ElectricCar has no constructor, while its superclass Car requires a double argument f
- ☐ d. the final AUTONOMY variable needs to be initialised
- ☐ e. the toString method in ElectricCar cannot access kmDriven, because it is declared as

What will be displayed when running the following code?

```
class SuperClass {
        public SuperClass() {
                System.out.println("Constructor from SuperClass");
                methodCall();
        }
        public void methodCall() { System.out.println("Method from Supe
}
class SubClass extends SuperClass {
        public SubClass() { System.out.println("Constructor from SubCla
        public void methodCall() { System.out.println("Method from SubC
}
public class Client {
        public static void main(String[] args) {
                SubClass s = new SubClass();
        }
}
```

- a. Constructor from SuperClass
  Constructor from SubClass
  Method from SuperClass
- b. Constructor from SuperClass
  Constructor from SubClass
  Method from SubClass
- c. Constructor from SubClass
  Constructor from SuperClass
  Method from SuperClass
- d. Constructor from SuperClass
  Method from SuperClass
  Constructor from SubClass
- e. Constructor from SuperClass
  Method from SubClass
  Constructor from SubClass


An abstract class:
a. may not contain abstract methods
b. may not be instantiated
c. is required to contain at least one abstract method
d. may be instantiated
e. that contains abstract methods needs each of its subclasses to either implemer

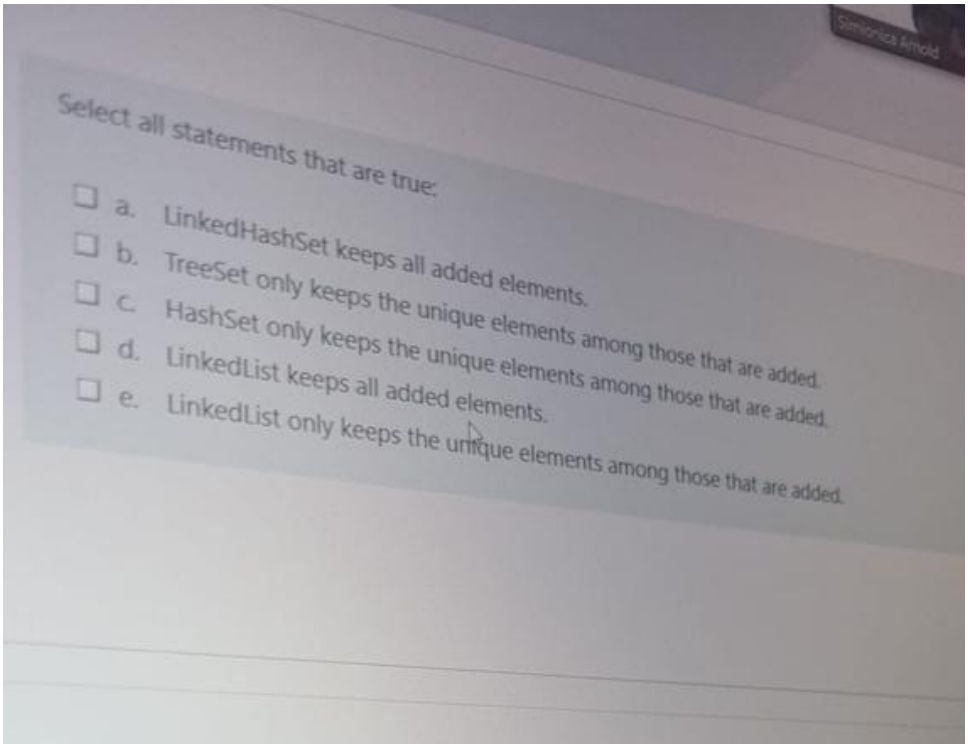

Question 6
Not yet answered
Marked out of 0.50
Flag question
Polymorphism is based on:
a. Method Overloading
b. Type Inheritance
c. Late Binding
d. Encapsulation
e. Early Binding

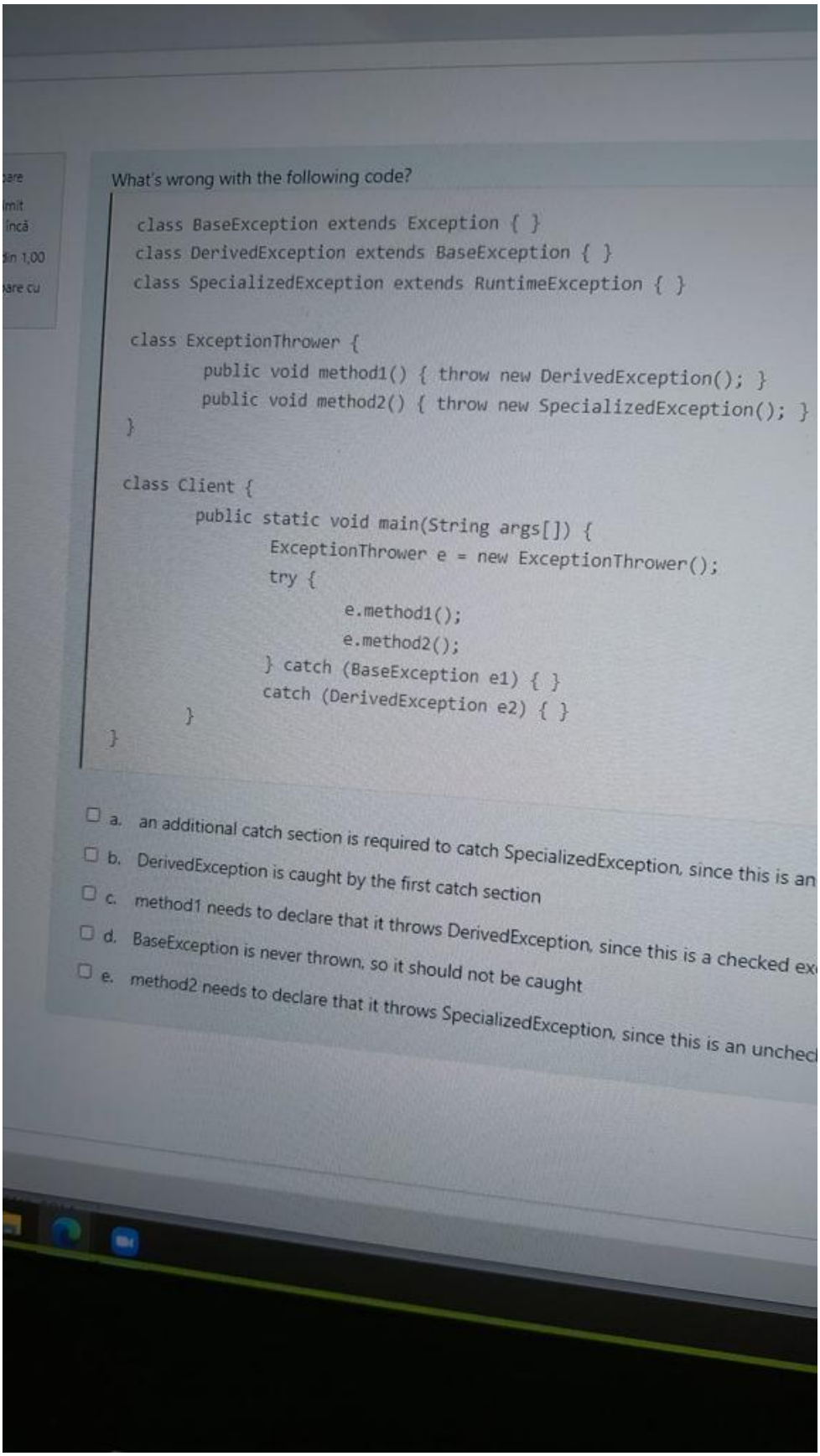
How many items in the format "Age: <number>" will be displayed and why?

```
import java.util.HashSet;
class Student {
        private int age;
        public Student(int age) { this.age = age; }
        public boolean equals(Object o) {
                return ((o instanceof Student) && (age == ((Student)o).age)); }
        public String toString() { return "Age: " + age; }
}
public class Client {
        public static void main(String args[]) {
                Student s1 = new Student(22);
                Student s2 = new Student(23);
                Student s3 = new Student(22);
                Student s4 = new Student(22);

                HashSet<Student> set = new HashSet<Student>(3);
                set.add(s1); set.add(s2); set.add(s3); set.add(s4);
                System.out.println(set);
        }
}
```

- ☐ a. 2 because the HashSet only stores the unique values for the age
- ☐ b. 3 because of new HashSet<Student>(3)
- ☐ c. 2 because there are only two different values and the Student class has the equals method overridden accordingly
- ☐ d. 4 because the equals method is overloaded, not overridden, so it is not considered
- ☐ e. 4 because the hash code default implementation provides different values, regardless of the age data member of


Select all statements that are true:
a. LinkedHashSet keeps all added elements.
b. TreeSet only keeps the unique elements among those that are added.
c. HashSet only keeps the unique elements among those that are added.
d. LinkedList keeps all added elements.
e. LinkedList only keeps the unique elements among those that are added.



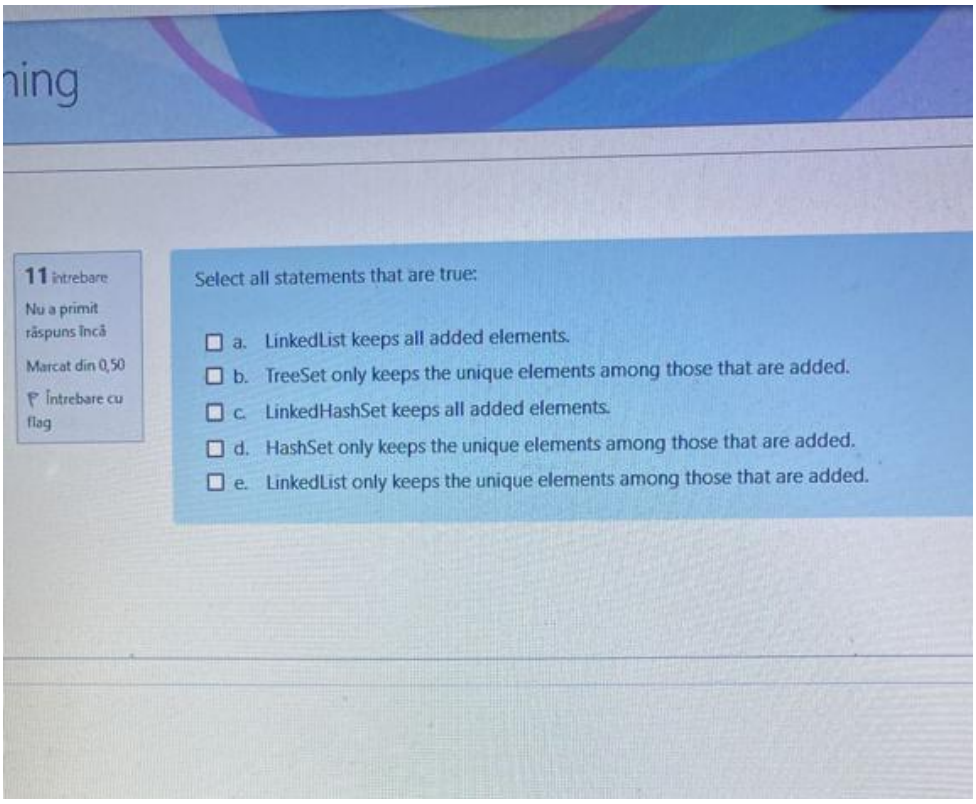
What's wrong with the following code?

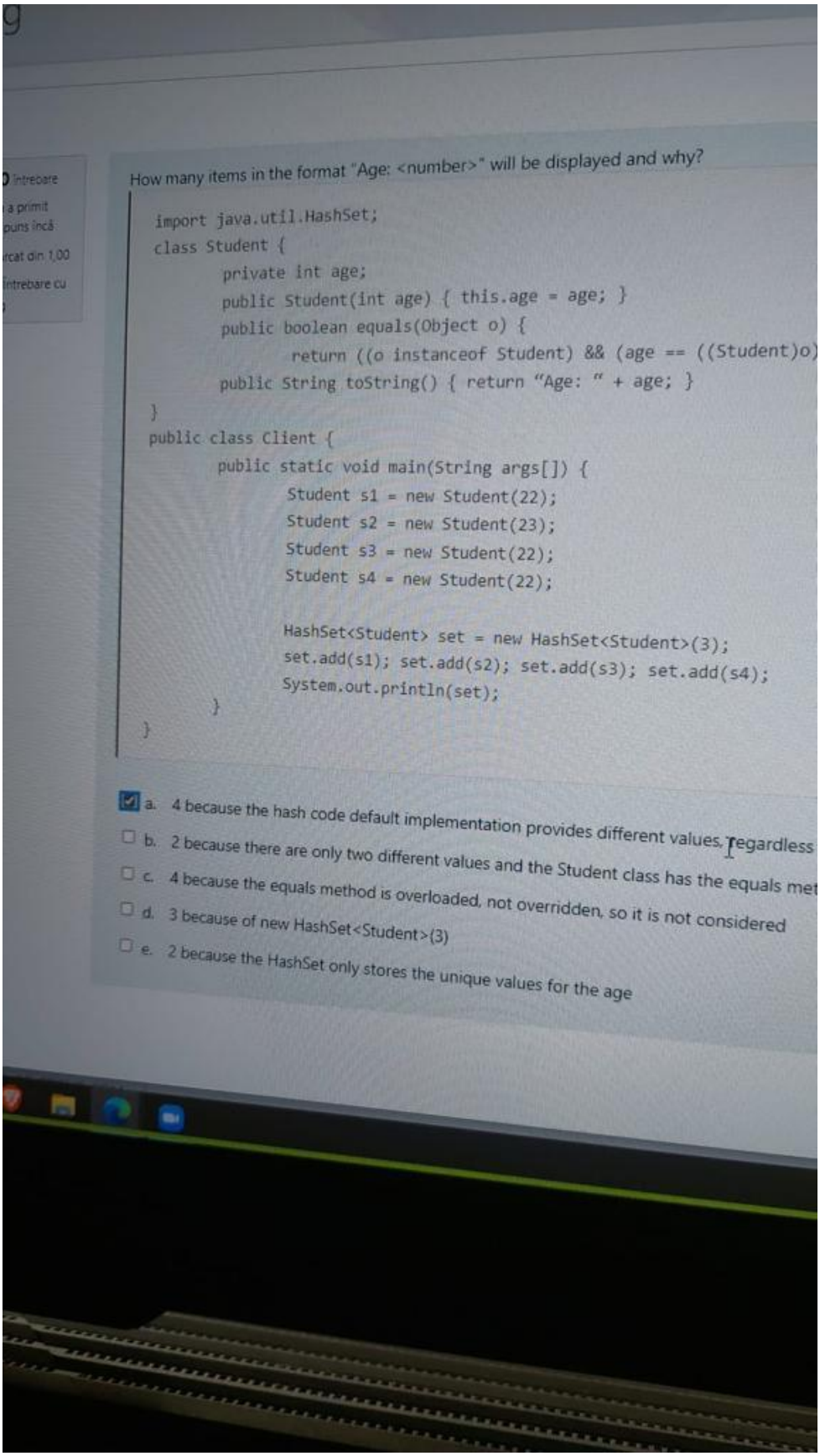
```
class BaseException extends Exception { }
class DerivedException extends BaseException { }
class SpecializedException extends RuntimeException { }

class ExceptionThrower {
        public void method1() { throw new DerivedException(); }
        public void method2() { throw new SpecializedException(); }
}

class Client {
        public static void main(String args[]) {
                ExceptionThrower e = new ExceptionThrower();
                try {
                        e.method1();
                        e.method2();
                } catch (BaseException e1) { }
                catch (DerivedException e2) { }
        }
}
```

- a. an additional catch section is required to catch SpecializedException, since this is an
- b. DerivedException is caught by the first catch section
- c. method1 needs to declare that it throws DerivedException, since this is a checked ex
- d. BaseException is never thrown, so it should not be caught
- e. method2 needs to declare that it throws SpecializedException, since this is an unchec


ning
11 întrebare
Nu a primit răspuns încă
Marcat din 0,50
Întrebare cu flag
Select all statements that are true:
a. LinkedList keeps all added elements.
b. TreeSet only keeps the unique elements among those that are added.
c. LinkedHashSet keeps all added elements.
d. HashSet only keeps the unique elements among those that are added.
e. LinkedList only keeps the unique elements among those that are added.



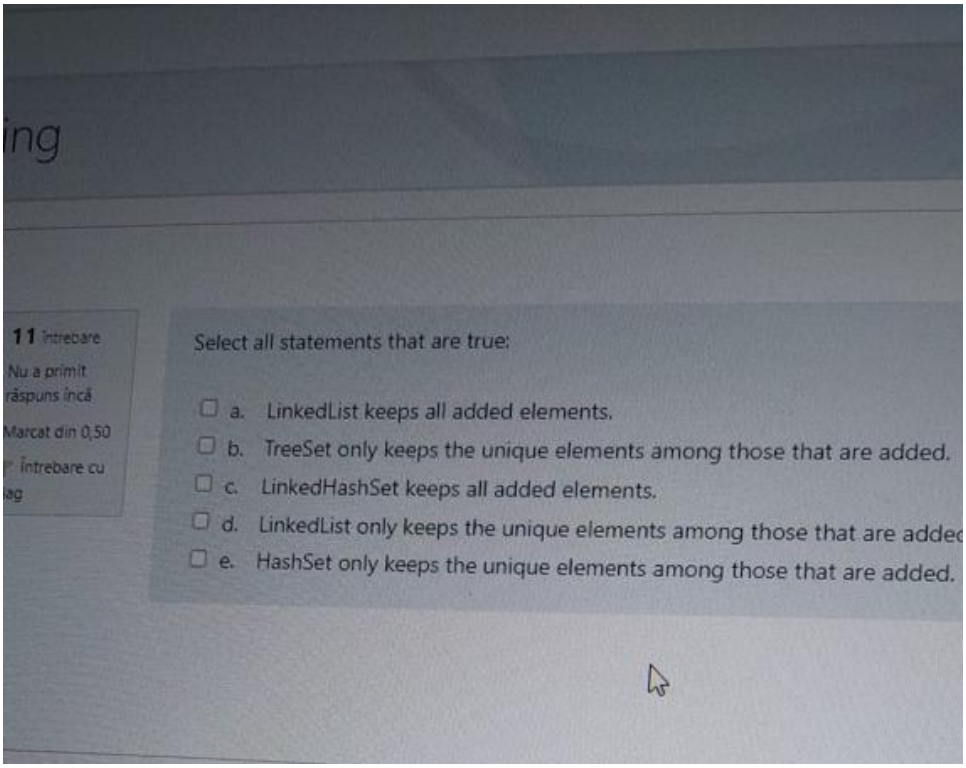
Întrebare
a primit
puns încă
rcat din 1,00
întrebare cu

How many items in the format "Age: <number>" will be displayed and why?

```
import java.util.HashSet;
class Student {
        private int age;
        public Student(int age) { this.age = age; }
        public boolean equals(Object o) {
                return ((o instanceof Student) && (age == ((Student)o)
        public String toString() { return "Age: " + age; }
}
public class Client {
        public static void main(String args[]) {
                Student s1 = new Student(22);
                Student s2 = new Student(23);
                Student s3 = new Student(22);
                Student s4 = new Student(22);

                HashSet<Student> set = new HashSet<Student>(3);
                set.add(s1); set.add(s2); set.add(s3); set.add(s4);
                System.out.println(set);
        }
}
```

- [x] a. 4 because the hash code default implementation provides different values, regardless
- [ ] b. 2 because there are only two different values and the Student class has the equals met
- [ ] c. 4 because the equals method is overloaded, not overridden, so it is not considered
- [ ] d. 3 because of new HashSet<Student>(3)
- [ ] e. 2 because the HashSet only stores the unique values for the age




Question **6**

Correct

Mark 0.50 out of 0.50

Flag question

Edit question

Polymorphism is based on:

- a. Method Overloading
- b. Late Binding
- c. Early Binding
- d. Encapsulation
- e. Type Inheritance

Your answer is correct.

The correct answers are: Type Inheritance, Late Binding

Make comment or override mark

Response history

For better code readability, the following should be avoided:

- [ ] a. generic names
- [ ] b. clear and concise names
- [ ] c. specific names
- [x] d. comments on constants
- [x] e. abstract names

The correct answers are:
generic names,

abstract names

Make comment or override mark

## Response history

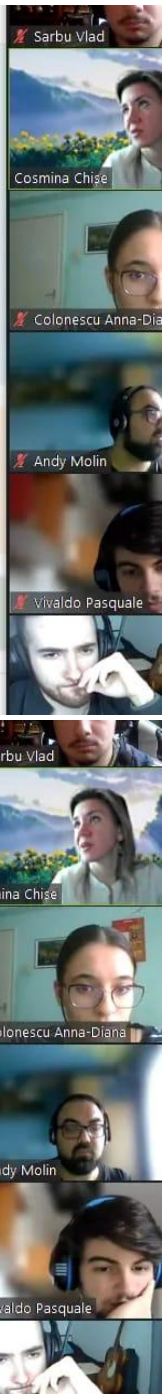
An abstract class:

- [x] a. may not contain abstract methods
- [x] b. that contains abstract methods needs each of its subclasses to either implement all abstract methods o abstract themselves (the subclasses)
- [ ] c. may be instantiated
- [x] d. may not be instantiated
- [ ] e. is required to contain at least one abstract method

The correct answers are:
may not be instantiated,

that contains abstract methods needs each of its subclasses to either implement all abstract methods or be decl themselves (the subclasses)

Make comment or override mark



Question **3**

Incorrect

Mark 0.00 out of 0.50

Flag question

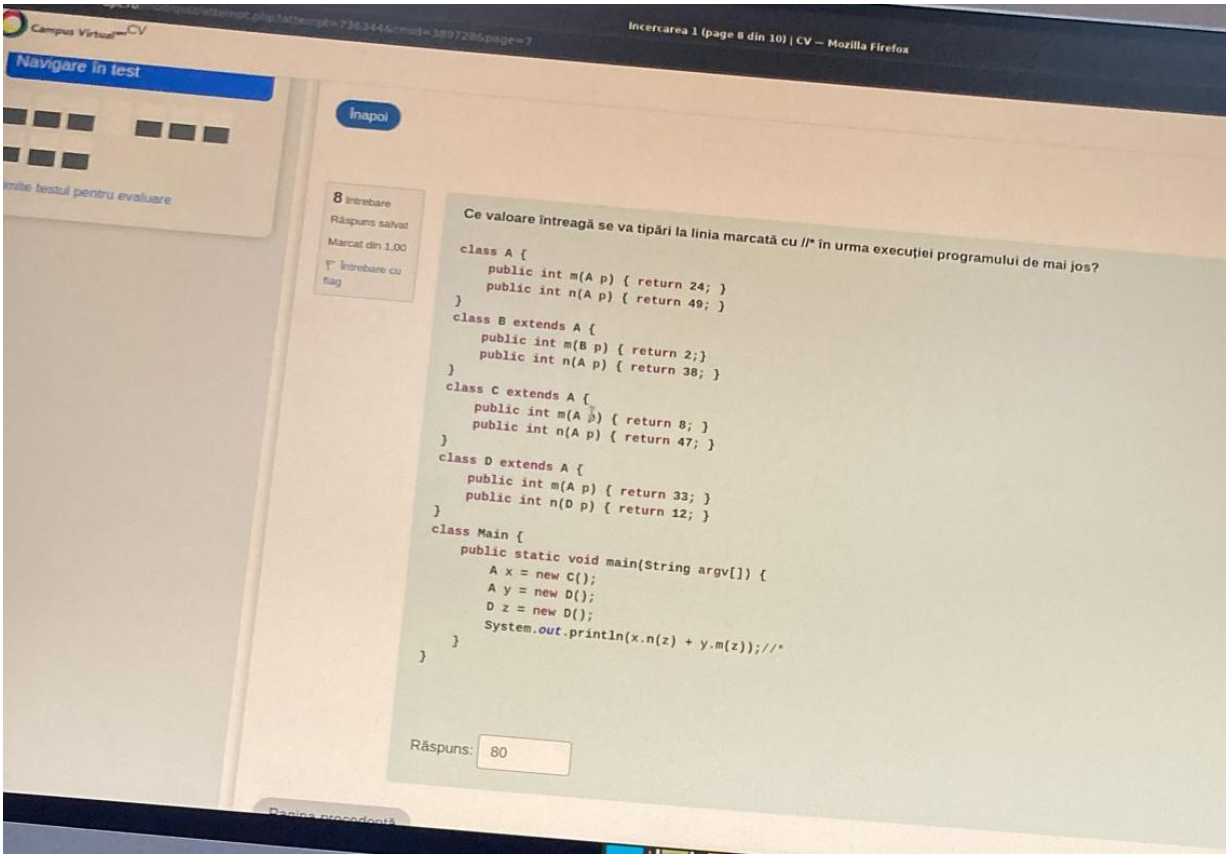
Edit question

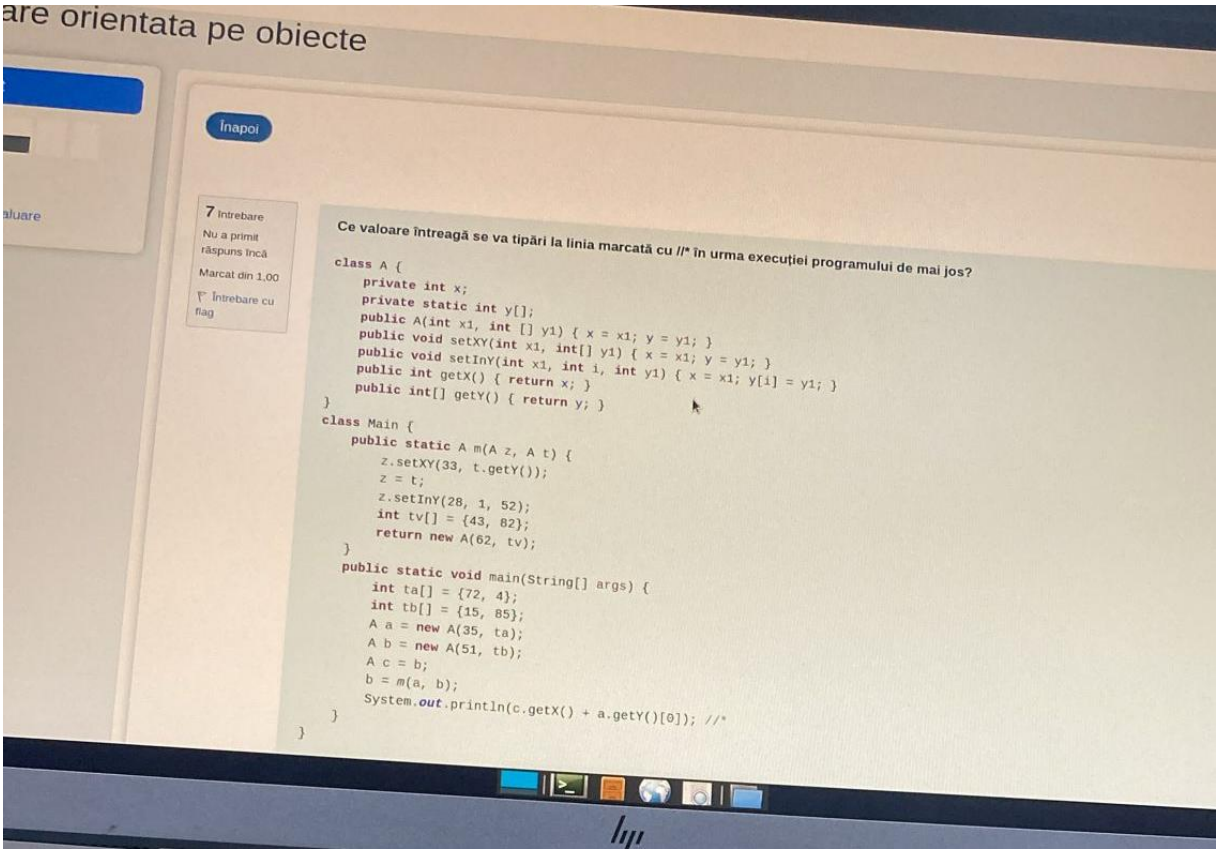
Identify the type of methods present in the following code:

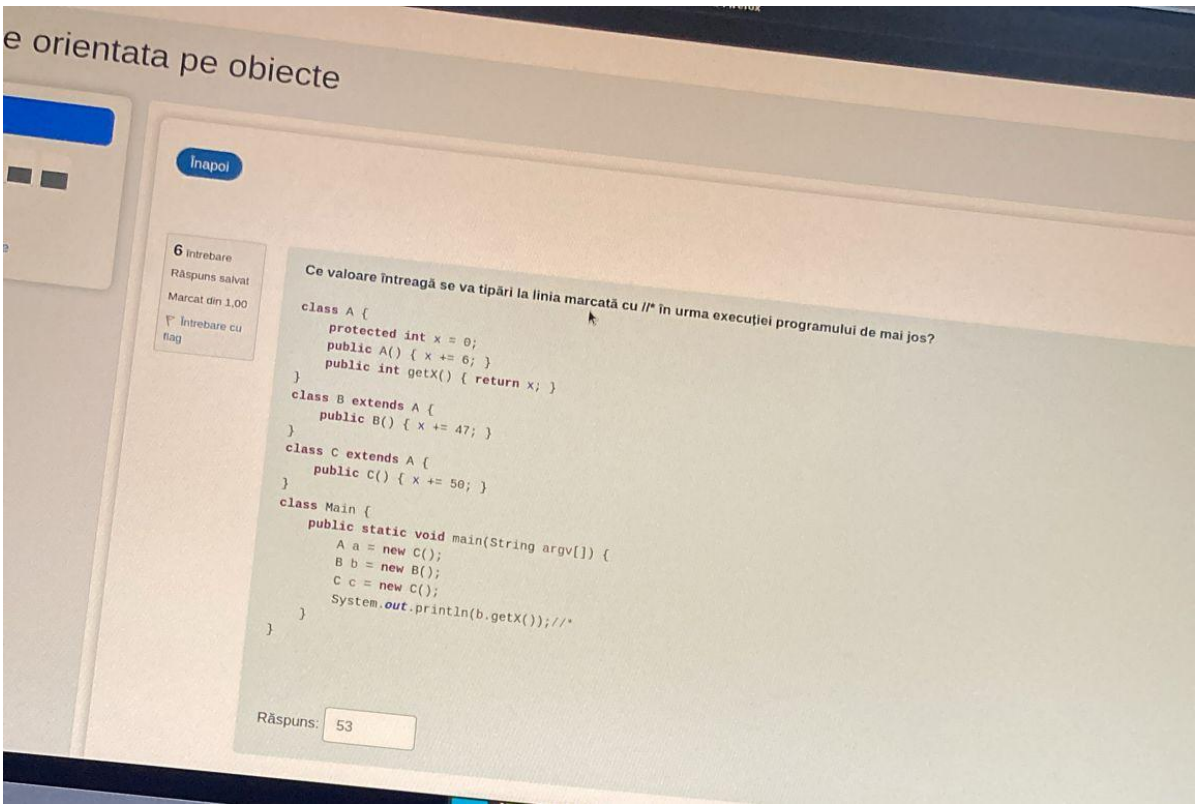
```
public class Car {
        private double kmDriven;
        public Car() {
                this.kmDriven = 0;
        }
        public Car(double kmDriven) {
                this.kmDriven = kmDriven;
        }
}
```

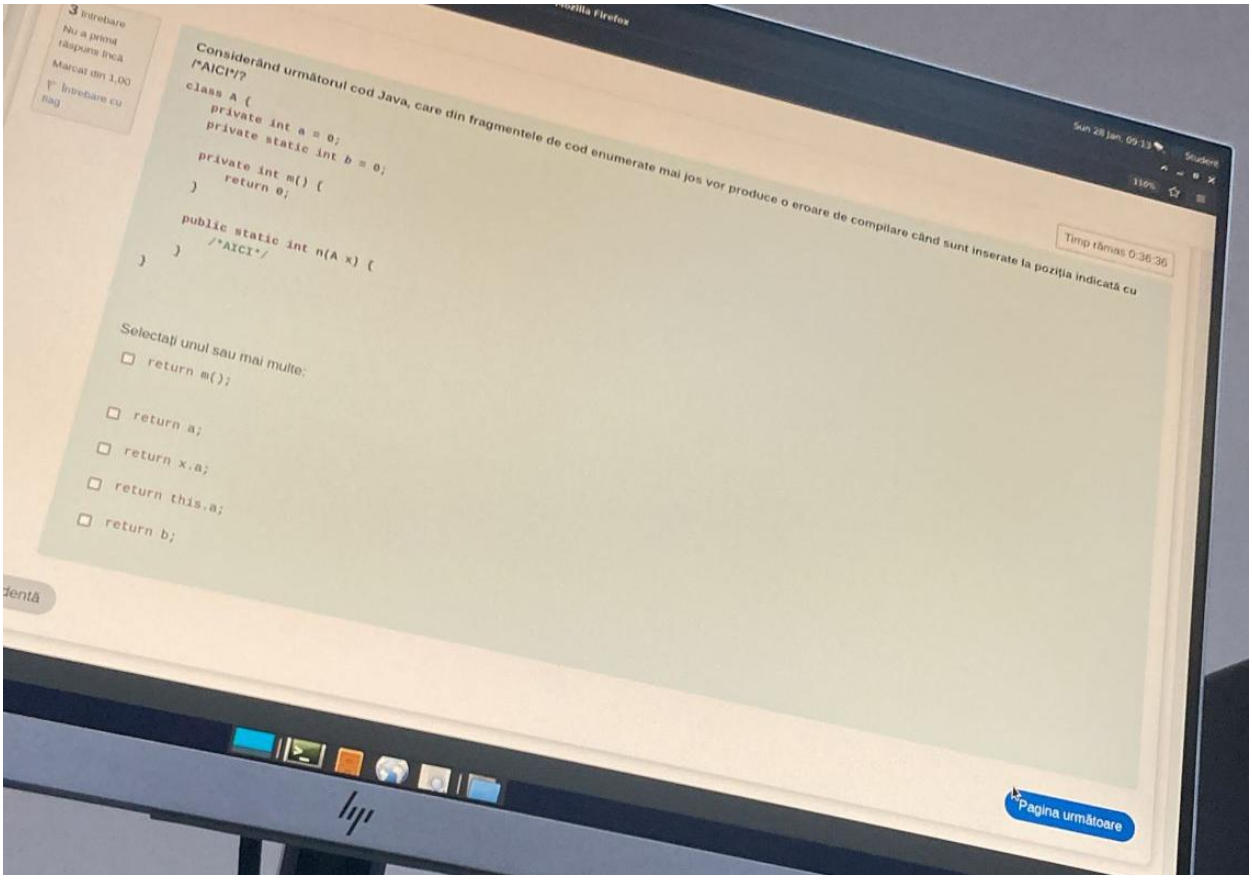
- [x] a. constructor
- [ ] b. overloaded
- [x] c. accessor: setter
- [ ] d. accessor: getter
- [ ] e. overridden

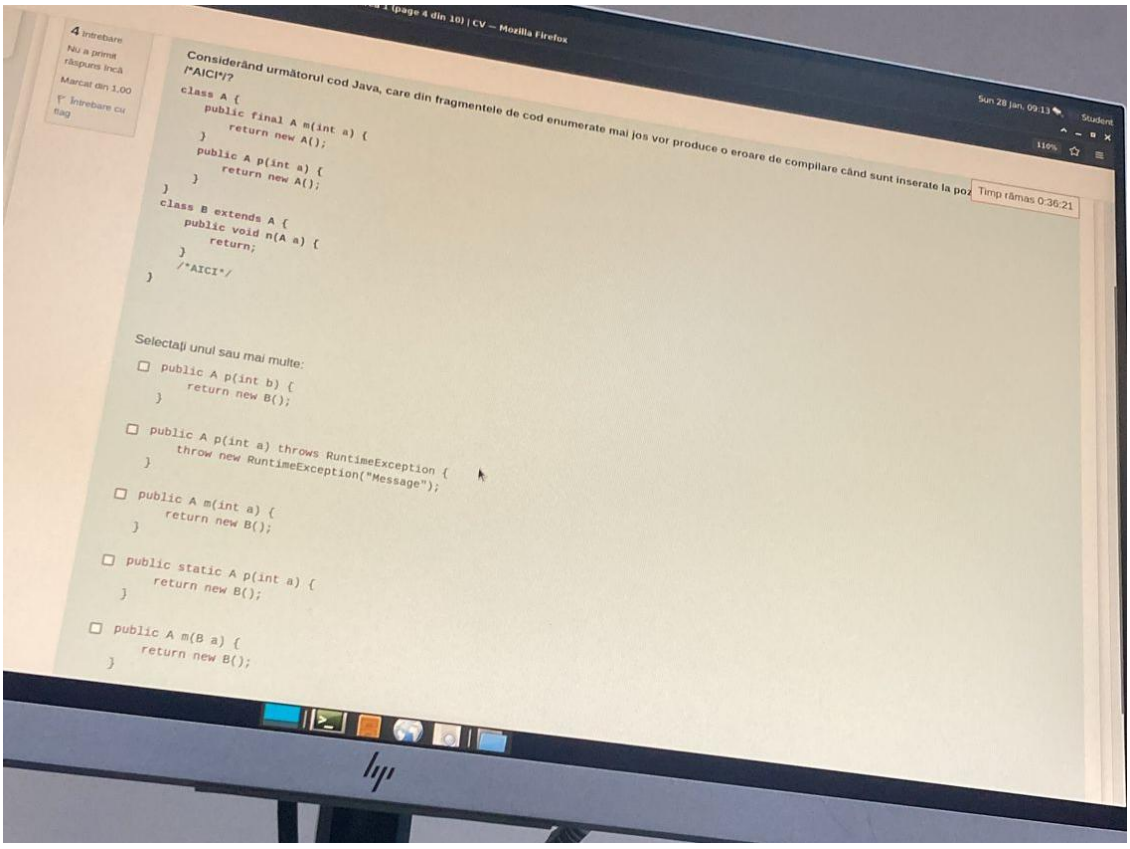
The correct answers are: overloaded, constructor

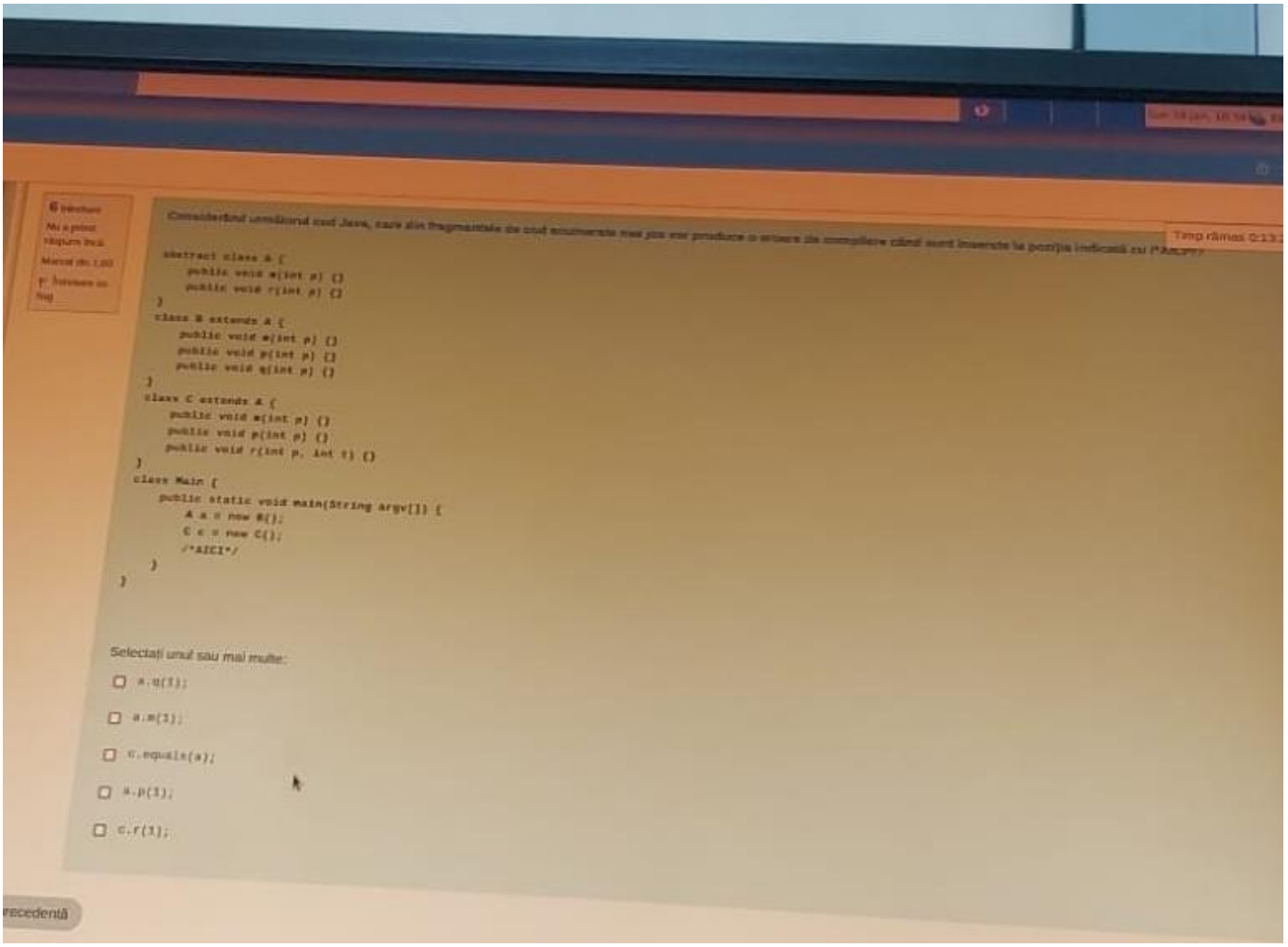



are orientata pe obiecte
Înapoi
7 Intrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag
Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?
class A {
private int x;
private static int y[];
public A(int x1, int [] y1) { x = x1; y = y1; }
public void setXY(int x1, int[] y1) { x = x1; y = y1; }
public void setInY(int x1, int i, int y1) { x = x1; y[i] = y1; }
public int getX() { return x; }
public int[] getY() { return y; }
}
class Main {
public static A m(A z, A t) {
z.setXY(33, t.getY());
z = t;
z.setInY(28, 1, 52);
int tv[] = {43, 82};
return new A(62, tv);
}
public static void main(String[] args) {
int ta[] = {72, 4};
int tb[] = {15, 85};
A a = new A(35, ta);
A b = new A(51, tb);
A c = b;
b = m(a, b);
System.out.println(c.getX() + a.getY()[0]); //*
}
}


e orientata pe obiecte
Înapoi
6 Intrebare
Răspuns salvat
Marcat din 1,00
Întrebare cu flag
Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?
class A {
protected int x = 0;
public A() { x += 6; }
public int getX() { return x; }
}
class B extends A {
public B() { x += 47; }
}
class C extends A {
public C() { x += 50; }
}
class Main {
public static void main(String argv[]) {
A a = new C();
B b = new B();
C c = new C();
System.out.println(b.getX());//*
}
}
Răspuns:
53


3 întrebare
Nu a primit răspuns încă
Marcat din 1,00
Întrebare cu flag
Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*AICI*/?
class A {
private int a = 0;
private static int b = 0;
private int m() {
return 0;
}
public static int n(A x) {
/*AICI*/
}
Timp rămas 0:36:36
Selectați unul sau mai multe:
return m();
return a;
return x.a;
return this.a;
return b;
Pagina următoare


4 întrebare
Nu a primit răspuns încă
Marcat din 1.00
Întrebare cu flag
Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poz
/*AICI*/?
class A {
public final A m(int a) {
return new A();
}
public A p(int a) {
return new A();
}
}
class B extends A {
public void n(A a) {
return;
}
/*AICI*/
}
Timp rămas 0:36:21
Selectați unul sau mai multe:
public A p(int b) {
return new B();
}
public A p(int a) throws RuntimeException {
throw new RuntimeException("Message");
}
public A m(int a) {
return new B();
}
public static A p(int a) {
return new B();
}
public A m(B a) {
return new B();
}



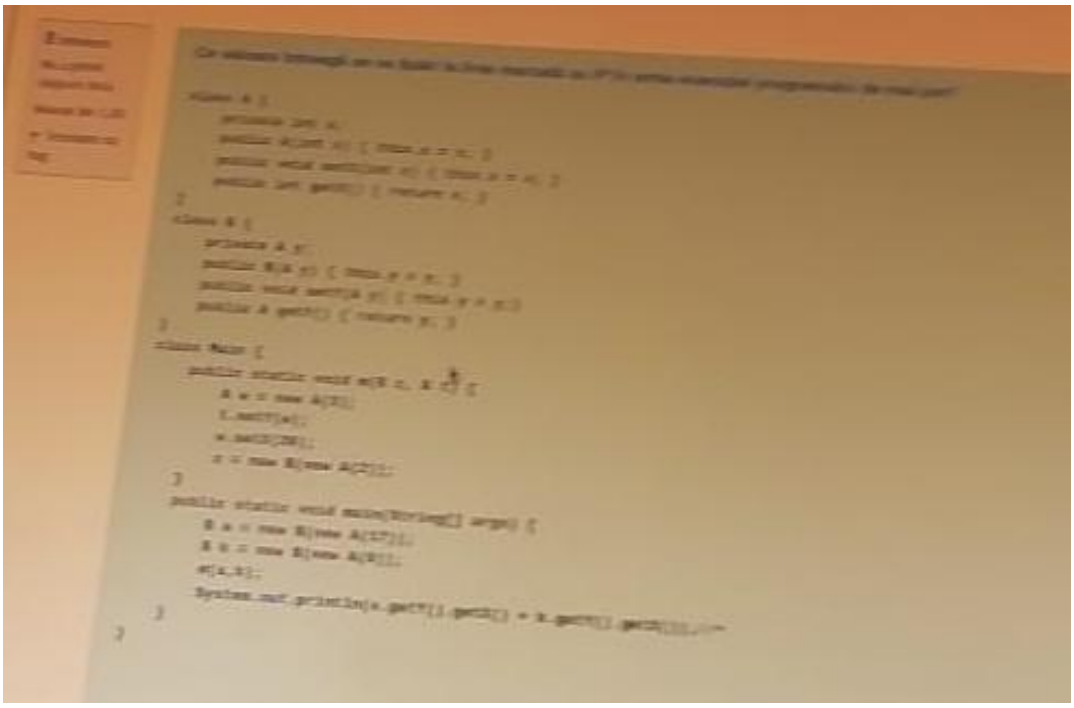
Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poziția indicată cu /*AICI*/?

```
abstract class A {
   public void m(int p) {}
   public void r(int p) {}
}
class B extends A {
   public void m(int p) {}
   public void p(int p) {}
   public void q(int p) {}
}
class C extends A {
   public void m(int p) {}
   public void p(int p) {}
   public void r(int p, int t) {}
}
class Main {
   public static void main(String Argv[]) {
      A a = new B();
      C c = new C();
      /*AICI*/
   }
}
```

Selectați unul sau mai multe:

- [ ] a.q(1);
- [ ] a.m(1);
- [ ] c.equals(a);
- [ ] a.p(1);
- [ ] c.r(1);

recedentă



**9** întrebare

Răspuns salvat

Marcat din 1,00

Întrebare cu flag

**Ce valoare întreagă se va tipări la linia marcată cu //* în urma execuției programului de mai jos?**

```
class Ex extends RuntimeException {}
class Main {
    public static void m(int x) throws Ex {
        if(x == 0) throw new Ex();
    }
    public static void main(String argv[]) {
        int a = 4;
        for(int i = 0; i < 3; i++) {
            try {
                m(i);
                a = a + 2;
            } catch(Ex e) {
                a++;
            } finally {
                a++;
            }
        }
        System.out.println(a);//*
    }
}
```

Răspuns: 10

Nu a primit răspuns încă

Marcat din 1,00

Întrebare cu flag

CV — Mozilla Firefox

Sun 28 Jan, 09:13

Timp rămas 0:37:

**Considerând următorul cod Java, care din fragmentele de cod enumerate mai jos vor produce o eroare de compilare când sunt inserate la poz /*AICI*/?**

```
interface I {
    public void g();
}
interface J extends I {
    public void h();
}
abstract class A implements I {
    public void m() {}
}
class B extends A implements J {
    public void p() {}
    public void g() {}
    public void h() {}
}
class Main {
    public void doSomething(I i, J j) {
        A a = new B();
        B b = new B();
        /*AICI*/
    }
}
```

Selectați unul sau mai multe:

- [ ] `b.m();`
- [ ] 
```
if (i instanceof B) {
    i.m();
}
```
- [ ] `((A)i).p();`
- [ ] `((B)j).m();`
- [ ] `((J)a).h();`