

1. Care va fi valoarea variabilei *counter* dupa executia blocului de instructiuni alaturat ?

- a) 1
- b) 2
- c) 3
- d) 4
- e) 5

```
int counter=0;
try {
    Objectobj[]=new Object[10];    counter++;
    obj[0] = new String("hello"); counter++;
    counter++;
}
catch(Exception e) { counter++; }
finally { counter++; }
```

2. Care vor fi erorile depistate la compilarea clasei alaturate ?

- a) Variabila *x* nu poate fi apelata dintr-un context static
- b) Variabila *x* nu poate fi apelata dintr-un contex ne-static
- c) Variabila *y* nu poate fi apelata dintr-un context static
- d) Variabila *y* nu poate fi apelata dintr-un context ne-static
- e) Metoda *g* nu are modificador de acces

```
class A {
    private int x=0;
    protected static int y=0;
    public void f() { x=1; y=1; }
    static void g() { x=2; y=2; }
}
```

3. Care este numarul de octeti ce vor fi folositi pentru memorarea valorilor variabilelor membre ale vectorului de obiecte `A a[]=new A[100];` unde clasa *A* este definita alaturi:

- a) 100
- b) 417
- c) 711
- d) 714
- e) 2100

```
class A {
    final int N=0;
    static double PI=3.14;
    static char c[] = {'a', 'b', 'c'};
    byte c[] = {1, 2, 3};
}
```

4. Care din afirmatiile de mai jos sunt adevarate ?

- a) Variabila *a* este definita corect
- b) Variabila *b* este definita corect
- c) Variabila *c* este definita corect
- d) Definitia metodei *inc* este gresita
- e) Corpul metodei *inc* este gresit

```
interface Test { int N=0; void inc(); }
abstract class AbstractImpl
    implements Test {}
class TestImpl extends AbstractImpl {
    void inc() { N++; }
    public static void main(String args[]) {
        Test a = new Test();
        TestImpl b = new AbstractImpl();
        AbstractImpl c = new TestImpl();
    }
}
```

5. Ce se va intampla la compilarea fisierului *Stiva.java* ce contine clasele alaturate ?

- a) Compilarea va fi cu succes si vor rezulta fisierele *ExceptieStiva.class* si *Stiva.class*
- b) Compilarea va fi cu succes, dar la executie metodele *push* si *top* pot fi sursele unor exceptii netratate
- c) Clasa *ExceptieStiva* nu va fi compilata deoarece nu supradefineste constructorul clasei *Exception*
- d) Clasa *Stiva* nu va fi compilata deoarece metodele *push* si *top* nu trateaza exceptiile ce pot aparea.
- e) Clasa *Stiva* nu va fi compilata deoarece metoda *pop* este incorect definita

```
class ExceptieStiva extends Exception {}

public class Stiva {
    Object v[] = new Object[100]; int top=0;
    public void push(Object x) { v[top]=x;
        top++; }
    public void pop(){
        if (top == 0) throw new ExceptieStiva();
        top --;
    }
    public Object top() { return v[top-1]; }
}
```

6. Care din urmatoarele afirmatii referitoare la appleturi sunt adevarate:

- a) Trebuie obligatoriu sa extinda clasa *java.applet.Applet*
- b) Trebuie obligatoriu sa supradefineasca metodele *init*, *start*, *stop*, *destroy*.
- c) Clasele unui applet pot fi grupate in mai multe pachete
- d) Un applet trebuie sa defineasca cel putin doua fire de executie
- e) Daca appletul nu se gaseste intr-o arhiva jar, clasele sale sunt transferate una cate una in conexiuni HTTP diferite.

7. Care din afirmațiile următoare referitoare la fire de execuție în Java sunt adevărate ?

- a) Orice fir de execuție este o instanță a clasei *Thread*
- b) Orice clasă care descrie un fir de execuție trebuie să implementeze obligatoriu interfața *Runnable*
- c) Pentru a controla accesul mai multor fire de execuție la o resursă comună, trebuie să declaram clasa ce descrie resursa respectivă folosind modificatorul *synchronized*
- d) Un fir de execuție poate să cedeze procesorul altor fire de execuție cu aceeași prioritate, chiar dacă nu și-a terminat încă activitatea.
- e) Singura modalitate de a opri un fir de execuție este prin apelul metodei *stop*

8. Care din afirmațiile următoare referitoare la programare Java în rețea sunt adevărate ?

- a) O aplicație Java poate deschide o conexiune cu un URL oarecare și citi informațiile de la acea adresă
- b) Un port este un număr reprezentat pe 16 biți
- c) Într-o aplicație client-server, clasa care descrie serverul trebuie obligatoriu să creeze câte un fir de execuție pentru fiecare client
- d) Comunicarea prin datagrame stabilește o conexiune sigură prin care sunt transmise pachete de informații
- e) Există posibilitatea de a transmite simultan mesaje de la server către un grup prestabilit de clienți

9. Scrieți o aplicație care citește un șir de caractere de la tastatură și îl scrie, în format binar (folosind serializarea), într-un fișier cu numele 'output.dat'.

10. Scrieți o aplicație Java care primește un număr întreg impar N ca argument de la linia de comandă și afișează la consolă un romb format din caracterul * și având diagonalele formate fiecare din N caractere *. Exemplu pentru $N=7$, se obține:

```
*
***
*****
*****
*****
***
*
```

NOTA. La subiectele 1-8 pot exista mai multe variante corecte (care se vor bifa).