LABORATOR1

1.Care dintre următoarele atribuiri sunt greșite din punct de vedere sintactic (vor provoca eroare la compilare) ?

a. **byte a = 128;**
b. short b = 128 * 129;
c. long c = 0xC;
d. double d = (float) 10E2;

OBS! Compilatorul își dă seama că 128 este prea mare pentru a fi reținut într-un byte. Dar nu face calculul efectiv 128 * 129 pentru verifica dacă poate fi reținut într-un short.

2. Care dintre următoarele instrucțiuni de import sunt greșite din punct de vedere sintactic ?

a. import java.lang.Object;
b. import java.*;
c. **import package java.awt;**
d. import static java.sql.Types.*;

3. Este corectă secvența de mai jos? Dacă da, care va fi rezultatul afișat? Motivați răspunsul.

```
String s1 = Integer.parseInt(1);
String s2 = Integer.parseInt(2);
System.out.println((s1+s2) == new String("12"));
```

Secvența este greșită și va produce eroare la compilare deoarece metoda parseInt() trebuie să primească un String și returnează un int - invers față de cum este scris.

4. Declarați și instanțiați o matrice M, cu trei linii și două coloane, cu elementele M(i, j)=10*i+j, într-o singură instrucțiune, fără a folosi structuri repetitive.

int M[][] = {{0, 1}, {10, 11}, {20, 21}};

sau

int M[][] = new int[][] {{0, 1}, {10, 11}, {20, 21}};

5. Care dintre următoarele variante nu este corectă?

a. Float a[] = new Float[10];
b. **Byte b[10] = new Byte[];**
c. Integer c[] = {10, 20};
d. **String d[] = {'x', 'y', 'z'};**

6. Ce va afișa secvența de mai jos?

```
int a,b;
for (a = 10, b = 1; a-- > ++b ; ) ;
System.out.println((a--) + "," + (++b));
```

a. 4,5
b. 5,6
c. **5,7**
d. 6,7

7. Care dintre următoarele variante nu este corectă?

a. **char []a = {"1", "2"};**
b. **double b[4] = new double[];**
c. int c[][] = {{1, 2}, {3, 4}};

8. Ce va afișa la execuție programul alăturat?

```
public class Asignare {
    public static void main(String args[]) {
        int a = 3;
        int b = (a = 2) * a;
       int c = b * (b = 5);
        System.out.println("a = " + a + ", b = " + b + ", c = " + c);
    }
}
```

Raspuns : a = 2, b = 5, c = 20

9. Ce se poate spune despre subprogramul Java alăturat?

```
int i = 1, suma = 0;
for ( ; ; ) {
    suma += i++;
    if (i > 5) break;
}
System.out.print(suma);
```

a. Eroare la compilare: lipsesc părțile componente ale lui for
b. Subprogramul se compilează și la execuție afișează 0
c. Subprogramul se compilează și la execuție afișează 10
d. **Subprogramul se compilează și la execuție afișează 15**

10. Care vor fi valorile variabilelor a, b și c după execuția secvenței de mai jos?

```
int a = 0, b = 1, c = (++a >= b-- ? a ++ : ++b);
```

a. a=2, b=1, c=1
b. a=1, b=0, c=1
c. a=2, b=1, c=0
d. **a=2, b=0, c=1**

LABORATOR 2

1. Care afirmații legate de metoda hashCode() a clasei Object sunt false?
   a. Trebuie să returneze un număr întreg;
   b. **Trebuie să returneze numere diferite pentru obiecte diferite;**
   c. **Nu poate fi supradefinită;**
   d. **Nu poate fi supraîncărcată.**

2. Care afirmații legate de metoda equals() a clasei Object sunt false?
   a. **Trebuie obligatoriu corelată cu metoda hashCode();**
   b. **Trebuie obligatoriu supradefinită;**
   c. Metoda acceptă orice obiect ca argument;
   d. Metoda poate fi aplicată pentru tablouri cu valori primitive.

3. Ce se va întâmpla la încercarea de a compila fișierul Test.java, descris mai jos?

```
class A extends Object {}

class B extends A {
private class C extends B {}
}
```

   a. Nu va fi compilat deoarece... (completați)
   b. **Va fi compilat și vor fi create unitățile de compilare... (completați)**
   c. A.class, B.class, B$C.class

4.Ce se va întâmpla la încercarea de a compila fișierul Test.java, descris mai jos?

```
public class A { static int A = 1; }

   class B extends A {
    B() { A = 2; }
}
```

RASPUNS :Eroare la compilare. Vă las pe voi să descoperiți de ce :)

5.Explicați pe scurt cum poate fi implementat șablonul de proiectare *Immutable Object* și dați un exemplu din API-ul standard Java.
RASPUNS :Câmpuri private, inițializate doar din constructor. Se expun doar getter-i, nu și setter-i. Dacă se implementează operații, aceastea ar trebui să creeze o nouă instanță și să o returneze, nu să modifice instanța curentă.
Exemplu din API-ul standard Java: String.

6.Ce puteți spune despre programul de mai jos?

```
class A {
      A(int x) { System.out.println("apel constructor A"); } }

class B extends A {
      B() { System.out.println("apel constructor B"); } }

public class C {
      public static void main(String args[]) { B b = new B(); } }
```

a. **Nu va fi compilat;**
b. Va afișa "apel constructor B";
c. Va afișa "apel constructor A";
d. Va afișa "apel constructor B" și apoi "apel constructor A";

7.Ce semnificație are cuvântul cheie super folosit ca metodă?

a.Apelează constructorul corespunzător ca număr de argumente al clasei;

b.Apelează constructorul corespunzător ca număr de argumente al superclasei;

c.Face referire la variabilele de instanță ale superclasei;

d.Cuvâtntul cheie super nu poate fi folosit ca metodă.

8. Ce puteți spune despre programul alăturat?

```
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 și va afișa la execuție: Mesajul Unu din C1, Mesajul Doi din C1

b.programul este corect și va afișa la execuție: Mesajul Unu din C1, Mesajul Doi din C2

c.programul este corect și va afișa la execuție: Mesajul Unu din C2, Mesajul Doi din C1

d.programul este corect și va afișa la execuție: Mesajul Unu din C2, Mesajul Doi din C2

e.va apărea eroare la compilare deoarece în clasa Test variabila obiect nu aparține clasei C2

9. Ce puteți spune despre prgramul alăturat?

```
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 și va afișa la execuție: 5 4

b.programul este corect și va afișa la execuție: 5.0 4

c.programul este corect și va afișa la execuție: 4.0 4

d.va apărea eroare la compilare deoarece în clasa C2 s-a suprascris greșit atributul x din clasa C1

e.va apărea eroare la compilare deoarece metoda suprascrisă f() din clasa C2 întoarce un tip diferit de void

10. Ce va afișa la execuție aplicația CtorDemo?

```
class A{
    int a=1;
    A() { a=2; }
}

class B extends A {
    int b=3;
    B() { b=a; }
}

public class CtorDemo {
    public static void main(String args[]) {
        B bobj = new B();
        System.out.println(bobj.b);
    }
      }
```

a.1
b.2
c.3
d.Nu se va afișa nimic deoarece codul este incorrect

LABORATOR 3

1. Care dintre următoarele variante nu este corectă?

a. **Writer a = new Writer("file.txt");**
b. Writer b = new PrintWriter(new FileWriter("file.txt"));
c. Writer c = new FileWriter("file.txt");
d. **BufferedWriter d = new BufferedWriter("file.txt");**

2.Ce puteți spune despre subprogramul alăturat?

```
double x = 0;
if (Double.isInfinite(2 / x))
    System.out.println("infinit");
else


 System.out.println("2 / 0");
```

a. Eroare la compilare datorită împărțirii la zero
b. Eroare la execuție datorită împărțirii la zero (se aruncă o excepție ArithmeticException)
c. **Programul este corect și va afișa: infinit**
d. Programul este corect și va afișa: NaN

3.Presupunem că avem un fragment de cod Java dintr-o aplicație care are drept de scriere în directorul de lucru curent și că în directorul curent nu există fișierul numit "fisier.txt". Care va fi rezultatul compilării și execuției codului alăturat?

```
1:  try {
2:    RandomAccessFile f1 = new RandomAccessFile("fisier.txt", "rw");
3:    BufferedOutputStream f2 = new BufferedOutputStream(f1);
4:    DataOutputStream f3 = new DataOutputStream(f2);
5:    f3.writeDouble(Math.PI);
6:    f3.close();
7:    f2.close();
8:    f1.close();
9:  } catch(IOException e) { }
```

a. **Codul Java de mai sus nu se poate compila;**
b. Codul Java de mai sus se poate compila, dar aruncă o excepție la linia 3;
c. Codul Java de mai sus se poate compila și executa dar nu are nici un efect pe sistemul de fișiere local;
d. Codul Java de mai sus se poate compila și executa, creându-se fișierul "fișier.txt" în directorul curent.

4. Ce se va întâmpla la execuția secvenței alăturate?

```
int test = 0, infinit = 0;
try {
    infinit = 1/0;
    test ++;
}
catch(Exception e) { test ++; }
finally { test ++; }
System.out.println("test=" + test + " infinit=" + infinit);
```

a. Se va afișa: test=2 infinit=NaN
b. **Se va afișa: test=2 infinit=0**
c. Se va afișa: test=1 infinit=0
d. Nu se va afișa nimic, programul se va opri cu excepție

5.Ce se va afișa la execuția următoarei secvențe de cod:

```
RuntimeException a = new RuntimeException();
System.out.println(a.getClass().getSuperclass().getSuperclass());
```

a. class java.lang.Exception
b. class java.lang.Error
c. **class java.lang.Throwable**
d. class java.lang.Object

6. Fie codul de mai jos. Care este rezultatul execuției lui?

```
1: File f1 = new File("numeDirector");
2: File f2 = new File(f1, "numeFisier");
```

a. eroare la compilare la linia 2, deoarece avem un apel recursiv nepermis
b. programul este corect și se creează în directorul curent un director cu numele numeDirector, iar în acest director se crează fișierul cu numele numeFisier
c. programul este corect și se creează în directorul curent un director cu numele numeDirector și fișierul cu numele numeFisier.
d. programul este corect și se creează în directorul curent doar fișierul cu numele numeFisier
e. **programul este corect, dar nu se creează nici un director și nici un fișier**

7. Care dintre afirmațiile de mai jos sunt adevărate?

a. când se creează o instanță a lui File și nu se folosește semantica de denumire a fișierelor de pe mașina locală, atunci constructorul va arunca o excepție IOException
b. când se creează o instanță a lui File și fișierul respectiv nu există pe sistemul de fișiere local, atunci se creează un astfel de fișier
c. când o instanță a lui File este eliberată de garbage collector, atunci fișierul corespunzător de pe sistemul local este șters
d. **nici una dintre afirmațiile de mai sus nu este adevărată**

8. Ce se va întâmpla la execuția următoarei secvențe?

```
float a=0;
int   b=0, test=0;
try {
   a=1/a; test ++;
   b=1/b; test ++;
}
catch(Exception e) { test ++; }
finally { test ++; }
System.out.println("test=" + test + " a=" + a + " b=" + b);
```

a. Nu se va afișa nimic, prograul se va opri cu excepție
b. Se va afișa: test=2 a=Infinity b=Infinity
c. **Se va afișa: test=3 a=Infinity b=0**
d. Se va afișa: test=4 a=0 b=0

9. Care dintre următoarele variante nu este corectă?

a. FileWriter a = new FileWriter("file.txt");
b. Writer b = new PrintWriter(new FileWriter("file.txt"));
c. PrintWriter c = new PrintWriter("file.txt");
d. **BufferedWriter d = new BufferedWriter("file.txt");**

10. Care este superclasa tutor claselor ce descriu excepții sau erori în Java?

a. **Throwable**
b. Exception
c. RuntimeException
d. Error

LABORATOR 4

1. O aplicaţie de gestiune a studenţilor unei universităţi conţine clasa Student şi câte o structură de date List<Student> studenti; cu studenţii fiecărei facultăţi. Care din invocările a) Arrays.sort(studenti) sau **b) Collections.sort(studenti)** este corectă și în ce condiții?

RASPUNS:Clasa Student trebuie să implementeze interfața Comparable și metoda compareTo().

2. În cadrul unei aplicații doriți să utilizați o clasă pentru lucru cu matrici rare cu numele org.math.cool.SparseMatrix, clasă aflată în arhiva coolmath.jar. Care sunt pașii pe care trebuie să îi faceți pentru a realiza acest lucru?

RASPUNS:Se adaugă arhiva în classpath. Pot fi menționați și pașii specifici IDE-ului pe care il folosiți.

3. O aplicație se conectează la un serviciu Web de unde citește valorile cursurilor valutare la o anumită dată. Declarați și instanțiați, folosind tipuri generice, o colecție care să poată memora informațiile citite de la respectivul serviciu Web.

RASPUNS:HashMap<String, Double> map = new HashMap<String, Double>();

4. Ce puteți spune despre programul alăturat?

```
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 și va afișa la execuție: 3 9 4**
b. va apărea eroare la compilare deoarece interfețele I1 și I2 conțin declarațiile aceloraşi atribute
c. va apărea eroare la compilare deoarece în interfețele I1 și I2 atributul i conține referință la atributul j
d. va apărea eroare la compilare deoarece interfețele I1 și I2 conțin declarațiile unui atribut de tip tablou
e. va apărea eroare la compilare deoarece atributele interfețelor I1 și I2 nu se pot accesa dintr-o metodă static

5. În ce condiții putem apela o metodă de sortare polimorfică unei colecții formate din obiecte de același tip?

a. **Metoda de sortare primește un argument de tip Comparator;**
b. Este suficientă supradefinirea metodei hashCode() în clasa care descrie obiectele;
c. **Clasa care descrie obiectele implementează interfața Comparable și metoda compareTo;**
d. Apelul este posibil în orice situație, fiind apelată metoda implicită equals() a clasei Object.

6. Să considerăm declarațiile alăturate? Care dintre următoarele variante nu este corectă?

```
class A{
    static int x = 37;
    transient int y = 47;
    static void f() {
        A aref = this;
        int i = x;
        int j = y;
        g();
    }
    void g() { }
}
```

a. **Declarația lui aref**
b. Declarația lui i
c. **Declarația lui j**
d. **Apelul metodei g()**

7. Să considerăm declarațiile alăturate de clase și interfețe. Care dintre acestea nu este corectă?

```
interface Test { }
abstract class AbstractImpl implements Test { }
class TestImpl extends AbstractImpl { }
```

a. **Test t = new Test();**
b. **TestImpl t = new AbstractImpl();**
c. **AbstractImpl t = new Test();**
d. AbstractImpl t = new TestImpl();

8. Ce puteți spune despre aplicația TestImpl?

```
public class TestImpl {
    public static void main(String[] args) {
        HelloImpl h1 = new HelloImpl(), h2 = h1;
        h1.message = "Salut";
        System.out.println(h2.sayHello());
    }
}
interface Hello {
    String sayHello();
}

class HelloImpl implements Hello {
    static String message = "Hello";

    String sayHello() {
        return message;
    }
}
```

a. Va afișa la execuție mesajul "Hello";
b. Va afișa la execuție mesajul "Salut";
c. Se va opri cu o excepție de tipul NullPointerException;
d. **Nu va fi compilată.**

9. Ce înțelegeți prin serializare?

a. **Scrierea unui obiect într-un fișier de unde să poată fi refăcut ulterior;**
b. Transformarea unui obiect într-un flux de octeți;
c. Adăugarea unui număr serial unui obiect pentru a putea fi deosebit de alte obiecte ale aceleiași clase, dar din alte versiuni;
d. Transmiterea obiectelor prin rețea.

10. Care dintre afirmațiile următoare referitoare la serializarea obiectelor nu este adevărată?

a. Pentru a fi serializabilă o clasă trebuie obligatoriu să implementeze una din interfețele Serializable sau Externalizable;
b. Obiectele serializate pot fi scrise atat in fișiere, cât și în memorie;
c. **Variabilele declarate cu modificatorii transient sau private nu participă la serializare.**
d. Mecanismul implicit de serializare a obiectelor unei clase poate fi modificat;
e. Clasele ObjectInputStream și ObjectOutputStream permit și serializarea tipurilor primitive de date.

LABORATOR 5

1. Care va fi valoarea variabilei s dupa execuția blocului de instrucțiuni alăturat?

```
int s = 0;
try {
    java.awt.Point p[] = new java.awt.Point[10];
    p[0].setLocation(-1, -1);
    s += 1;
} catch(IllegalArgumentException e) { s += 2; }
  catch(NullPointerException e) { s += 3; }
  finally{ s += 4; }
```

a. 0
b. 1
c. 5
d. 6
e. **7**

2. Ce se va întâmpla la compilarea fișierului Test.java ce conține clasele alăturate?

```
class A {

    double x = 0;

    A(double x) { this. x = x; }
    A(float x) { this. x = x; }
}

class B extends A {
    double y = 0;
    B(double x) {
        y = 0;
        super(x);
    }
}
```

a. Compilarea va fi cu succes.
b. Compilarea va fi cu succes, dar la execuție instanțierile obiectelor din clasa A pot provoca excepții.
c. Clasa B nu va fi compilată deoarece nu supradefinește ambii constructori.
d. **Clasa B nu va fi compilată deoarece constructorul este definit greșit.**
e. Clasa A nu va fi compilată deoarece al doilea constructor este greșit.

3. Care dintre afirmațiile următoare referitoare la colecții nu este adevărată?

a. Clasa TreeMap menține sortate elementele sale.
b. Singura modalitate de a parcurge o mulțime de tip HashSet, fără a folosi alte structuri de date, este prin intermediul iteratorilor.
c. Clasa Collection este superclasa tuturor claselor ce descriu colecții.
d. Într-un obiect de tip HashMap, cheia și valoarea sunt de tip Object.

e. **Orice colecție de elemente poate fi sortată folosind metoda sort() din clasa Collections.**

4.Scrieți metoda contains(Point p, Point p1, Point p2) care să testeze dacă un punct p se găsește pe dreapta determinată de punctele p1 și p2.

RASPUNS :

```
public static boolean contains(Point p, Point p1, Point p2) {
  return p.getY() - p1.getY() ==
    ((p.getX() - p1.getX()) * (p2.getY() - p1.getY())) / (p2.getX() - p1.getX());
}
```

5. În urma compilării fișierului Test.java a rezultat, pe lângă fișierul Test.class, și fișierul Test$1.class. Care este explicația acestui fapt?

a. În fișierul Test.java mai există o clasă cu numele Test$1.
b. Clasa Test conține o subclasă cu numele Test$1.
c. Clasa Test conține o clasă internă statică cu numele 1.
d. **Clasa Test conține o clasă anonimă.**
e. Test$1.class este un fișier temporar creat de către compilator.

6. Care dintre următoarele variante nu este corectă?

a. **Map a = new Map();**
b. Map b = new HashMap();
c. **SortedMap d = new HashMap();**
d. SortedMap c = new TreeMap();

7. Ce se va întâmpla la compilarea fișierului Stiva.java ce conține clasele alăturate?

```
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]; }
}
```

a. Compilarea va fi cu succes şi vor rezulta fişierele ExceptieStiva.class şi Stiva.class.
b. Compilarea va fi cu succes, dar la execuţie metodele push() şi top() pot fi sursele unor excepţii netratate.
c. Clasa ExceptieStiva nu va fi compilată deoarece nu supradefineşte constructorul clasei Exception.
d. Clasa Stiva nu va fi compilată deoarece metodele push() şi top() nu tratează excepţiile ce pot apărea.
e. **Clasa Stiva nu va fi compilată deoarece metoda pop() este incorect definită.**

8. În fişierul A.java sunt următoarele declaraţii de clase şi interfeţe:

```
interface X {}
interface Y {}
public class A implements X {}
public class B extends A implements X, Y {}
```

Care este motivul pentru care secvenţa de mai sus nu va fi compilată?

a. În Java nu există moştenire multiplă.
b. Clasa B face o dublă implementare interfeţelor.
c. **În fişier există două clase publice.**
d. Pentru a putea fi implementate, interfeţele trebuie să fie publice.

9. Care va fi valoarea variabilei tip după execuţia secvenţei alăturate?

```
int c = (int) 'b';
String tip = "nimic";
switch(c) {
    case 'a'-'z': tip = "litera";
    case '0'-'9': tip = "cifra"; break;
    default: tip = "altceva";
}
```

a. "litera"
b. "cifra"
c. **"altceva"**
d. "nimic"

10. Considerându-se declaraţia "public private int h;" să se precizeze care afirmaţie este adevărată:

a. variabila h va fi accesată în mod public deoarece se ia în considerare primul modificator de acces.
b. variabila h va fi accesată în mod private deoarece se ia în considerare ultimul modificator de acces.
c. **va fi eroare la compilare deoarece o variabilă nu poate fi în acelaşi timp accesată public şi private.**

LABORATOR 6

1.Ce se va întâmpla la compilarea următorului program:

```
public class SubiectLicentaUnu {
    int i = 3, j = 4;
    public static void main(String [] arg) {
        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 dun funcția main() nu putem accesa variabilele nestatice i și j**
c. va apărea eroare la compilare deoarece nu s-au inserat paranteze între operatorii ++, adică (i++) și (++j)
d. se va afișa 8
e. se va afișa 9

2. Ce se va afișa în cadrul următorului program (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 și 7 nu se vor compila deoarece lipsesc numele metodelor și tipurile returnate
b. linia 7 nu se va compila deoarece poate exista doar un inițializator static
c. codul se compilează și execuția produce ieșirea x = 15
d. codul se compilează și execuția produce ieșirea x = 10
e. **codul se compilează și execuția produce ieșirea x = 3**

3. Ce se afișează în urma execuției următorului program?

```
public class SubiectLicentaTrei {
    public static void main(String args[]) {
        double d = 12.34;
        ScadeUnitate su = new ScadeUnitate();
        su.scadeUnitate(d);
        System.out.println(d);
    }
}

class ScadeUnitate {
    public void scadeUnitate(double d) {
        d = d - 1.0;
    }
}
```

a. 0.0
b. -1.0
c. **12.34**
d. 11.34
e. eroare la compilare deoarece lipsește constructorul clasei ScadeUnitate

4. Ce rezultat obținem la execuția următorului cod Java:

```
int x = 1;
String[] sir = {"Test", " Licenta", " JAVA"};
sir[--x] += ".";
for (int i = 0; i < sir.length; i++)
System.out.print(sir[i]);
```

a. șirul: Test.
b. șirul: Test Licenta. JAVA
c. șirul: Tes
d. **șirul: Test. Licenta JAVA**
e. o excepție de tip ArrayIndexOutOfBoundsException

5. Considerându-se secvența de cod alăturată, să se precizeze care afirmație este adevărată:

```
i) float a = 2;
ii) float b = 1.3;
```

a. instrucțiunile i) și ii) sunt corecte
b. **instrucțiunea i) este corectă, instrucțiunea ii) este incorectă**
c. instrucțiunea i) este incorectă, instrucțiunea ii) corectă
d. instrucțiunile i) și ii) sunt incorecte

6. Ce puteți spune despre următorul subprogram Java?

```
String s1 = "anul" + 200 + 2,
s2 = 200 + 2 + "anul";
System.out.println("s1 = " + s1 + ", s2 = " + s2);
```

a. s1 = anul202, s2 = 202anul
b. **s1 = anul2002, s2 = 202anul**
c. s1 = anul202, s2 = 2002anul
d. s1 = anul2002, s2 = 2002anul
e. Eroare la execuție: este necesară o conversie explicită!

7. Ce puteți spune despre programul alăturat? (liniile sunt numerotate)

```
1.  public class TestTablouri {
2.      public static void main(String[]args) {
3.          int[] a = {2, 6};
4.          Object b = a;
5.          System.out.print(b[0] + " ");
6.          byte[] c = new byte[2];
7.          c = a;
8.          System.out.print(c[1] + " ");
9.      }
10. }
```

a. programul este corect și va afișa la execuție: 2 6
b. va apărea eroare de compilare doar la linia 4 deoarece b necesită o conversie explicită
c. va apărea eroare de compilare doar la linia 5 deoarece b[0] nu poate fi accesat
d. **vor apărea erori de compilare la linia 5 deoarece b[0] nu poate fi accesat și la linia 7 deoarece tipul componentelor tablourilor a și c nu sunt compatibile**
e. va apărea eroare la execuție din cauza liniei 5, aruncându-se excepția ClassCastException

8. Ce se poate spune despre programul alăturat?

```
public class Apel {
    static int f(short a) { return a * 2; }
    public static void main(String[] args) {
        System.out.println(f(3));
    }
}
```

a. **Eroare la compilare: nu există o definiție a metodei f() cu parametrul int**
b. Eroare la compilare: metoda f() nu întoarce o valoare de tip short
c. Eroare la compilare: metoda statică f() trebuie apelată astfel: Apel.f(3)
d. Programul se compilează și la execuție afișează 6

9. Ce se va afișa în urma execuției secvenței alăturate?

```
String s1 = new String("hello"),
s2 = "Hello".toLowerCase();
System.out.println(s1 == s2);
```

a. Secvența nu va fi compilată deoarece variabila s2 nu este instanțiată
b. Secvența nu va fi compilată deoarece șirurile nu pot fi comparate cu ==
c. **false, deoarece s1 și s2 au adrese de memorie diferite**
d. true, deoarece toLowerCase convertește un șir la litere mici
e. true, deoarece comparația șirurilor nu ține cont de litere mici sau mari

10. Care dintre următoarele afirmații referitoare la cele 4 linii numerotate ale metodei main este corectă?

```
public class Test {
    public static String sayHello() {
        return "Hello";
    }

    public static void main(String[] args) {
        Test t = new Test();
        String a = Test.sayHello();    // 1
        t.sayHello().toUpperCase();    // 2
        System.out.println(sayHello()); // 3
        args[0] = t.sayHello();        // 4
    }
}
```

a. Totul este corect, clasa va fi compilată
b. Linia 1 este greșită, deoarece metodele de clasă nu pot fi apelate astfel
c. Linia 2 este greșită, deoarece metoda sayHello returnează un șir de caractere
d. Linia 3 este greșită, deoarece metoda sayHello este statică
e. **Linia 4 este greșită, deoarece este posibil ca vectorul args să nu aibă elemente**

LABORATOR 8

1. Să considerăm declarația A a = new A(); unde clasa A este definită mai jos:

```
class A implements Runnable {
    int counter = 0;
    public void run() { while (true) counter++; }
}
```

Care dintre variantele de mai jos instanțiază și lanseaza un fir de execuție?

a. a.run();
b. a.start();
c. new Thread(a).run();
d. **new Thread(a).start();**

2. 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 declarăm 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** (vezi Thread Priorities, yield() and Join).
e. Singura modalitate de a opri un fir de execuție este prin apelul metodei stop().

3. Ce se va afișa la execuția următorului program?

```
1. public class Exemplu {
2.  public static void main(String args[]) {
3.      A.B a = new A().new B();
4.      a.f();
5.  }
6.}
7. class A {
8.  private int x = 4;
9.  public class B {
10.     private int y = 5;
11.     public void f() {
12.         System.out.println("x = " + x);
13.         x = 3;
14.         System.out.println("x = " + x + ", y = " + y);
15.     }
16. }
17.}
```

a. Eroare de compilare la linia 3 deoarece construcția A.B nu este validă.
b. Eroare de compilare la linia 4 deoarece metoda f() nu aparține clasei A.
c. Eroare de compilare la linia 9 deoarece nu se poate declara o clasă în interiorul altei clase.
d. Eroare de compilare la linia 12 deoarece nu se poate accesa data privată x.
e. **Se va afișa:**
   x = 4
   x = 3, y = 5

4. Scrieți o metodă care primeste ca argumente o colecție de numere întregi *v* precum și un alt întreg *a*. Metoda va elimina din colecție toate valorile strict mai mici decât *a* și va returna numărul de eliminări efectuate. Metoda poate să accepte atât colecții de tip ArrayList, cât și LinkedList.

OBS!Se punctează signatura funcției (tip returnat, parametri - în special cel de tip List<Integer>), parcurgerea listei fără folosirea iteratorilor (de ce?), compararea elementelor, eliminarea elementelor și numărarea eliminărilor.

VARIANTA SOLUTIE

```
 public static int elimina(List<Integer> v, int a) {
  int count = 0;
  for(int i = 0; i < v.size(); i++) {
    if(v.get(i).compareTo(a) < 0) {
      v.remove(i);
      i--;
      count++;
    }
}

  return count;
}
```

5. Care dintre următoarele afirmații sunt adevărate relativ la codul de mai jos?

```
1. public class Exemplu extends Thread {
2.    public void run() {
3.        System.out.print("Ultima ");
4.        yield();
5.        System.out.print("intrebare");
6.    }
7.    public static void main (String args[]) {
8.        Exemplu e = new Exemplu();
9.        e.start();
10.    }
11.}
```

a. Compilarea va eșua la linia 4 deoarece yield() se poate apela doar în cod sincronizat.
b. Compilarea va eșua la linia 4 deoarece yield() se poate apela doar într-un bloc try/catch.
c. Compilarea se va face cu succes. La execuție, nu se va afișa nimic.
d. Compilarea se va face cu succes. La execuție, se va afișa *Ultima*.
e. **Compilarea se va face cu succes. La execuție, se va afișa *Ultima intrebare*.**

LABORATOR 11

1. Ce definește structura de mai jos? Descrieți o modalitate de utilizare a ei.

```
@Target(ElementType.FIELD)
public @interface NotNull { }
```

RASPUNS :Structura definește o adnotare care poate fi folosită pe câmpuri. Dacă aveți astfel de întrebări, dați și exemplu
Ex: @NotNull
int x;

2. Implementați metoda drawShape care primește ca argument un String ce reprezintă numele unei clase ce implementează interfața java.awt.Shape și un obiect de tip java.awt.Graphics2D, funcționalitatea acesteia fiind desenarea respectivei forme geometrice (implicite), folosind metoda Graphics2D.draw().

Se punctează:

- Signatura metodei: void drawShape(String c, Graphics2D g)
- Folosirea mecanismului de reflection: z = Class.forName(c).newInstance();
- Desenarea: g.draw((Shape)z);

3. • Care dintre următoarele afirmații legate de applet-uri nu sunt corecte?

a. Un applet poate fi descărcat de pe un server Web sub formă de arhivă jar.
b. **Execuția unui applet începe întotdeauna cu metoda main().**
c. Applet-urile pot accesa fișiere de pe mașina client prin JNLP API

JNLP API, Accessing the Client Using JNLP API

d. Într-un applet pot fi folosite mai multe fire de execuție.

4. Fie a un obiect de tip java.awt.Applet ce implementează interfața Runnable. Care dintre următoarele variante va determina pornirea unui fir de execuție?

a. a.start();
b. new Thread(a).run();
c. **new Thread(a).start();**
d. Thread.start(a);

5. Care informații referitoare la applet-uri sunt corecte?

a. **Conțin obligatoriu o clasă ce extinde java.applet.Applet.**
b. **Pot fi formate din mai multe clase.**
c. **Clasele unui applet pot fi grupate în mai multe pachete.**
d. Folosesc cel puțin două fire de execuție.

6. Care din următoarele afirmații referitoare la applet-uri sunt adevărate?

a. **Trebuie obligatoriu să extindă clasa java.applet.Applet**
b. Trebuie obligatoriu să supradefinească metodele init(), start(), stop(), destroy().
c. **Clasele unui applet pot fi grupate în mai multe pachete.**
d. Un applet trebuie să definească cel puțin două fire de execuție.
e. Dacă applet-ul nu se găsește într-o arhivă .jar, clasele sale sunt transferate una câte una în conexiuni HTTP diferite.

7. Ce se poate spune despre programul alăturat?

```
public class Program {
    static int x = 6;
    public static void main(String[] args) {
        System.out.print("x = " + x);
        int x = (x = 3) * x;
        System.out.print(", x = " + x);
    }
}
```

a. Eroare la compilare: variabila x este declarată de două ori.
b. Programul se compilează și la execuție afișează *x = 6, x = 3*.
c. **Programul se compilează și la execuție afișează *x = 6, x = 9*.**
d. Programul se compilează și la execuție afișează *x = 6, x = 18*.

8. Care va fi valoarea variabilei counter după execuția blocului de instrucțiuni alăturat?

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

a. 1
b. 2
c. 3
d. **4**
e. 5

9. Care vor fi erorile depistate la compilarea clasei alăturate?

```
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; }
}
```

a. **Variabila x nu poate fi apelată dintr-un context static**
b. Variabila x nu poate fi apelată dintr-un context ne-static
c. Variabila y nu poate fi apelată dintr-un context static
d. Variabila y nu poate fi apelată dintr-un context ne-static
e. Metoda g nu are modificator de acces