1. Care variantă reprezintă o supraîncărcare corectă pentru metoda: protected int getGrade(String course)
   a) protected int getGrade(String course) throws IOException
   b) private int getGrade(String course)
   c) protected long getGrade(String course)
   d) public long getGrade(int studID)

2. Ce se afiseaza?

```
public class Test {
   public static void main(String []args) {
      Drink tea = new Tea();
      tea.make();
   }
}

class Drink {
   public static void make() {
      System.out.println("Making drink");
   }
}

class Tea extends Drink {
   public static void make() {
      System.out.println("Making tea");
   }
}
```

3. Ce se afiseaza?

```
public class BasicInit {
      private int x;
      private boolean flag;
      protected String s;
      @Override
      public String toString() {
                  return x + " " + flag + " " + s;
      }
      public static void main(String []args) {
                  BasicInit basicInit = new BasicInit();
                  System.out.println(basicInit);
      }
}
```

4. Ce înseamnă constructorul implicit (default)?
   a) constructor fără parametri declarat de utilizator
   b) constructor fără parametri adăugat de Java dacă nici un constructor nu a fost declarat
   c) constructor fără implementare
   d) constructor fără modificatori de access

5. Ce colecție ar fi mai eficientă de folosit dacă dorim să stocăm o secvență de elemente pe care să o modificăm rar dar pe care să o accesăm foarte des?
   a) LinkedList
   b) ArrayList
   c) Vector
   d) niciuna din variante

6. Ce se afișează?

```
Set<Integer> mySet = new LinkedHashSet<>();
mySet.add(1);
mySet.add(10);
mySet.add(100);
System.out.println(mySet);
```

   a) [10, 1, 100]
   b) [100, 10, 1]
   c) [1, 10, 100]
   d) numerele vor fi afișate într-o ordine arbitrară

7. Câte interfețe poate extinde o interfață în Java?
   a) una singura
   b) oricâte
   c) interfață implementează alte interfețe, nu le extinde
   d) niciuna

8. Care dintre următoarele colecții nu sunt iterabile (nu implementează patternul Iterable oferit în Java prin interfața Iterable)?
   a) ArrayList
   b) HashMap
   c) Queue
   d) Set

9. Ce afiseaza?

```
class Test {
   public static void main(String[] args) {
      System.out.println(breakingStuff());
   }

   public static int breakingStuff() {
```

```
        try {
          try {
            try {
              throw new Exception();
            } catch (Exception e) {
              return 1;
            } finally {
              return 2;
            }
          } catch (Exception e) {
            return 3;
          } finally {
            return 4;
          }
        }
    }
```

10. Ce va afisa?

```
class Test {
   public static void main(String args[]) {
      String s1 = "Wow, am luat 10 la grila la POO!";
      String s2 = new String(s1);
      System.out.println((s1 == s2) + " " + s1.equals(s2));
   }
}
```

11. Care afirmație despre LinkedHashSet din API-ul Java pentru colecții este adevarată?
    a) nu există clasa LinkedHashSet
    b) pastrează ordinea de inserare a elementelor și nu permite duplicate
    c) pastrează perechi de forma (Key, Value) și permite duplicate
    d) este o listă simplu înlănțuită unde fiecare element este o pereche (Key, Value)

12. Ce se va afisa?

```
public static void main(String[] args) {
 String s = 0+1+"ONE"
                          +3+2+"TWO"+"THREE"
                          +5+4+"FOUR"+"FIVE"+5;
 System.out.println(s);
}
```

13. Care afirmatie e falsa?
    a) interfața Comparable<T> conține o metodă int compareTo(T o);
    b) interfața Comparator<T> conține o metodă int compare(T o1, T o2);
    c) interfața Comparator<T> conține o metodă int compareTo(T o);

d) interfața Comparator<T> conține o metodă boolean equals(Object obj) ce face override metodei din Object;

14. Ce se afiseaza?

```
class A {
  int x;
  public A() { init(0); }
  protected void init(int x) { this.x = x; }
}

class B extends A {
  public B() { init (super.x + 1); }
  public void init(int x) { this.x = x + 1; }
}

public class Test {
  public static void main(String[] args) {
    A a = new B();
    System.out.println(a.x);
  }
}
```

15. Care din liniile de mai jos sunt corecte?

```
interface ITest {
  protected int x = 10;
  int y;
  int z = 20;
  abstract void foo();
  final int f(int x);
}
```

16. Ce se afiseaza?

```
class A {
        private int x = 5;

        private void hidden() {
                System.out.println(x);
        }

        public void show_hidden() {
                hidden();
        }
}
class B extends A {
        public int x = 10;

        public void hidden() {
                System.out.println(x);
        }
}
public class Main {
        public static void main(String[] args) {
                B b = new B();
                b.show_hidden();
        }
}
```

17. Care afirmație este adevărată în contextul limbajului Java?
    a) O clasă poate implementa oricâte interfețe și poate moșteni oricâte clase (abstracte sau concrete)
    b) O clasă poate implementa o singură interfață și poate moșteni oricâte clase (abstracte sau concrete)
    c) O clasă poate implementa oricâte interfețe și poate moșteni o singură clasă (abstractă sau concretă)
    d) O clasă poate implementa oricâte interfețe și poate moșteni oricâte clase abstracte și o singură clasă concretă

18. Care dintre afirmațiile următoare sunt adevărate in contextul limbajului Java?
    a) Dacă a.equals(b) == false, atunci a.hashcode()==b.hashcode() este false.
    b) Metoda equals trebuie implementată pentru a determina dacă două obiecte sunt egale.
    c) Chiar dacă propriile obiecte nu suprascriu equals, ele pot fi folosite drept chei în obiecte de tip Map, fără a avea vreun caz de funcționare incorectă.
    d) HashSet nu permite duplicate și nu menține ordinea elementelor. - 1, 2, 3 - 1, 4 - 2, 3, 4 + 2, 4