1. Fie urmatoarele definitii de clase si interfete:

```
interface I1 {}
interface I2 {}
interface I3 extends I1 {}
class C1 implements I1 {}
class C2 implements I2 {}
class C3 implements I3 {}
class C4 extends C3 implements I2 {}
```

   In fiecare caz specificati daca este corect sau daca apar erori (specificati daca sunt de compilare sau rulare) si de ce.

```
a) I3 b = new C3();
b) I2 d = new C4();
c) C4 e = new C3();
d) C4 f = (C4)(new C3());
e) I1 g1 = new C1(); C4 g2 = new C4(); g1 = g2;
```

2. Fie urmatoarele definitii de clase:

```
abstract class A {
      int bar(A a) { return 0; }
      int bar(B b) { return 1; }
      int bar(C c) { return 2; }
}

class B extends A {
      int bar(A a) { return 3; }
      int bar(B b) { return 4; }
      int bar(C c) { return 5; }
}

class C extends B {
      int foo() { return ((A)this).bar((A)this); }
}

class Test1
{
      public static void main(String[] args)
      {
            C x = new C();
            System.out.println(x.foo());
      }
}
```

   Ce se tipareste? De ce?

3. Fie urmatorul segment de cod:

```
class Exemplu {
      public static void main(String[] args) {
            LinkedList<?> s = new LinkedList<Integer>();
            Iterator<Object> x = s.iterator();
      }
}
```

   Apar erori de compilare/rulare? De ce?

4. Functia `Integer.parseInt(String s)` intoarce un `int` pentru valoarea memorata in `s`. Daca `s` nu contine un intreg, `parseInt` arunca o exceptie de tip `NumberFormatException`. Scrieti un segment de cod care memoreaza in variabila rez valoarea intoarsa de apelul `Integer.parseInt(unSir)` dar pune `1` in `rez` daca este aruncata exceptia `NumberFormatException`.

5. Care dintre urmatoarele interfete sunt interfete functionale? Explicati:

```
public interface Mult {
      int opMult(int a, int b);
}

public interface SpecMult extends Mult{
      int opMult(double a, double b);
}

public interface Ointerf {
}
```