Listagem A

```
class MinhaExcepcao extends Exception { }

class ListagemA {
    static void divide( int num, int denom)throws MinhaExcepcao {
        int resultado = 0;
       try {
            resultado = num/denom;
            System.out.print("try 1 * ");
            if (resultado == 1) throw new MinhaExcepcao();
            System.out.print("try 2 * ");
        } catch (ArithmeticException e) {
            System.out.print("catch 3 * ");
            throw e;
        }catch (MinhaExcepcao e) {
            System.out.print("catch 4 * ");
        } finally {
            System.out.print("finally 4 * ");
        }
         System.out.print("fim 5 * ");
    }

    public static void main(String[] args) {

       try {
            divide(4, 0);
            System.out.print("try main1 * ");
       } catch (Exception e) {
            System.out.print("catch main1 * " );
       } finally{
            System.out.print("finally main1 * ");
       }

       System.out.println();

       try {
            divide(4, 4);
            System.out.print("try main2 * ");
       } catch (Exception e) {
            System.out.print("catch main2 * ");
       } finally{
            System.out.print("finally main2 * ");
       }

       System.out.println();

       try {
            divide(4, 2);
            System.out.print("try main3 * ");
       } catch (Exception e) {
            System.out.print("catch main3 * ");
       } finally{
            System.out.print("finally main3 * ");
       }

       System.out.println();
    }

}
```

Listagem B

```
1  import java.util.ArrayList;
2  import java.util.HashSet;
3  import java.util.List;
4  import java.util.Set;
5
6  interface IAlimentar {
7      void alimentar();
8  }
9  interface IVoar {
10     void voar();
11 }
12 interface ICaminhar {
13     void caminhar();
14 }
15 abstract class SerVivo implements IAlimentar {
16     private String nome;
17
18     public SerVivo(String nome) {
19         this.nome = nome;
20     }
21     abstract public void qualHabitat();
22     @Override
23     public boolean equals( Object ob){
24         return ob instanceof SerVivo && ob.nome.equalsIgnoreCase(nome);
25     }
26     @Override
27     public int hashCode(){
28         return nome.hashCode();
29     }
30 }
31 class Morcego extends SerVivo implements IVoar{
32
33     public Morcego(String nome) {
34         super(nome);
35     }
36
37     @Override
38     public void voar() {
39         System.out.println(" voar de morcego ");
40     }
41     public void alimentar() {
42         System.out.println(" alimentar de morcego");
43     }
44     public void qualHabitat(){
45         System.out.println(" habitat de morcego");
46     }
47         @Override
48     public boolean equals( Object ob){
49         return ob instanceof IVoar;
50     }
51     @Override
52     public int hashCode(){
53         return 1;
54     }
55 }
56 class Avestruz extends SerVivo implements IVoar, ICaminhar{
57
58     public Avestruz(String nome) {
59         super(nome);
60     }
61
62     @Override
63     public void voar() {
64         System.out.println(" voar de avestruz ");
65     }
66     public void alimentar() {
67         System.out.println(" alimentar de avestruz");
68     }
69     public void qualHabitat(){
```

```
        System.out.println(" habitat de avestruz");
    }
    public void caminhar(){
        System.out.println(" habitat de avestruz");
    }
    @Override
    public boolean equals( Object ob){
        return super.equals(ob) && ob instanceof ICaminhar;
    }
}

class Teste {

    public static void main(String[] args) {
        IAlimentar a1 = new IAlimentar();
        SerVivo sv = new SerVivo("sss");

        IAlimentar m1 = new Morcego("aaa");
        m1.alimentar();
        m1.qualHabitat();
        m1.voar();

        SerVivo m2 = new Morcego("bbb");
        m2.alimentar();
        m2.qualHabitat();
        m2.voar();

        List<ICaminhar> caminhantes = new ArrayList<>();
        caminhantes.add( new Avestruz("ccc"));
        caminhantes.add( new Morcego("ddd"));

        System.out.println("A " + m1.equals(m2));

        SerVivo av = new Avestruz("aaa");
        System.out.println("B " + av.equals(new Avestruz("aaa")));
        System.out.println("C " + av.equals(new Avestruz("bbb")));
        System.out.println("D " + av.equals(m1));

        Set<Morcego> morcegos = new HashSet<>();
        morcegos.add(m1);
        morcegos.add(new Morcego("eee"));
        morcegos.add(new Morcego("fff"));
        morcegos.add(new Morcego("ggg"));
        System.out.println("Tem " + morcegos.size() + " elementos ");
    }
}
```

Listagem C

```
interface Constantes {
    public static final int DISPONIVEL = 1, COM_CARTAO_ADMITIDO = 2, COM_ACESSO_A_CONTA = 300,
EM_MANUTENCAO = 5;
}

class Conta {
    private int num;
    private int codigo;
    private int saldo;
    private int tentativas = 3;

    public Conta(int num, int codigo, int saldo) {
        this.num = num;
        this.codigo = codigo;
        this.saldo = saldo;
    }

    public int getCodigo() {
        return codigo;
    }
    public int getSaldo() {
        return saldo;
    }
    public boolean temTentativas() {
        return tentativas > 0;
    }
     public void gastaTentativa(){
        --tentativas;
    }
    public void repoeTentativas(){
        tentativas = 3;
    }
    public boolean levantar(int quantia) {
        if (saldo >= quantia) {
            this.saldo -= quantia;
            return true;
        }
        return false;
    }
    // outros métodos
    // . . .
}

class Multibanco implements Constantes {
    private Map<Integer, Conta> contas = new HashMap<>();
    private Conta conta = null;
    private int situacao = DISPONIVEL;

    public Multibanco() {
        // carregar contas
       //  ….
    }

    public void insereCartao(int numCartao) {
        if (situacao == DISPONIVEL) {
            conta = contas.get(numCartao);
            if (conta != null) { //existe uma conta correspondente ao cartao inserido
                situacao = COM_CARTAO_ADMITIDO;
            }
        }
    }
```

```
    public void digitaCodigo(int cod) {
        if (situacao == COM_CARTAO_ADMITIDO && conta.temTentativas()) {
            if (cod == conta.getCodigo()) {
                situacao = COM_ACESSO_A_CONTA;
                conta.repoeTentativas();
            } else { // digitou codigo errado
                conta.gastaTentativa();
                if(!conta.temTentativas()){
                    situacao = DISPONIVEL;
                }
            }
        }
    }
    public void fazLevantamento(int quantia) {
        if (situacao == COM_ACESSO_A_CONTA) {
            conta.levantar(quantia);
        }
    }
    public String consultaSaldo() {
        if (situacao == COM_ACESSO_A_CONTA) {
            return "Saldo: " + conta.getSaldo();
        } else {
            return "Sem acesso";
        }
    }
    public void retiraCartao() {
        if (situacao == COM_CARTAO_ADMITIDO || situacao == COM_ACESSO_A_CONTA) {
            situacao = DISPONIVEL;
        }
    }
    public void fazManutencao() {
        if (situacao == DISPONIVEL) {
            situacao = EM_MANUTENCAO;
        }
    }

    public void terminaManutencao() {
        if (situacao == EM_MANUTENCAO) {
            situacao = DISPONIVEL;
        }
    }
}
```

Listagem D

```
class Conta {
    private int num;
    private int codigo;
    private int saldo;
    private int tentativas = 3;

    public Conta(int num, int codigo, int saldo) {
        this.num = num;
        this.codigo = codigo;
        this.saldo = saldo;
    }

    // outros métodos
    // . . .


     @Override
    public String toString() {
        /* A */
    }
     @Override
    public int hashCode() {
        /* B */
    }

    @Override
    public boolean equals(Object obj) {
        /* C */
    }
}


class ContaGold extends Conta {

    private /* D */ somaDosSaldosIniciais = 0;

    public ContaGold(int num, int codigo, int saldo) {
        /* E */
    }

    public boolean levantar(int quantia) {
        //  procedimento especial
        return true;
    }

    public /* F */ getSomaDosSaldosIniciais(){
        return  somaDosSaldosIniciais;
    }

    @Override
    public String toString() {
        /* G */
    }
}

class FabricaContas {

    public static final int CONTA_BASICA = 0, CONTA_GOLD = 1;

    static Conta criaConta(int tipo, int num, int codigo, int saldo ){
        /* H */
    }
}
```

```
public class UsaContas{

    public static void main(String args[]) {
        List <Conta> contas = new ArrayList<>();

        System.out.println("Soma dos saldos iniciais das contas gold: " +
                                                          ContaGold.getSomaDosSaldosIniciais());

        Conta conta1 = FabricaContas.criaConta(FabricaContas.CONTA_BASICA, 1, 1111, 200);
        Conta conta2 = FabricaContas.criaConta(FabricaContas.CONTA_GOLD, 2, 2222, 200);
        Conta conta3 = FabricaContas.criaConta(FabricaContas.CONTA_GOLD, 3, 3333, 40000);
        Conta conta4 = FabricaContas.criaConta(FabricaContas.CONTA_GOLD, 4, 4444, 30000);

        if(conta1 != null)   contas.add(conta1);
        if(conta2 != null)   contas.add(conta2);
        if(conta3 != null)   contas.add(conta3);
        if(conta4 != null)   contas.add(conta4);

        for(Conta c:contas){
            System.out.println(c);
        }
        System.out.println("Soma dos saldos iniciais das contas gold: " +
                                                          ContaGold.getSomaDosSaldosIniciais());
    }
}
```

Listagem E

```
class ModeloObservavel /* A */ {

    private /* B */  multibanco;

    public ModeloObservavel(/* B */ multibanco) {
        this.multibanco = multibanco;
    }

    public String consultaSaldo() {
        /* C */
    }

    public void fazLevantamento(int quantia) {
        /* D */
        /* E */
        /* F */
    }
}
```

Listagem F

```
class VistaConta extends JPanel /* A */{
    private ModeloObservavel modelo;

    private JLabel saldo;
    private JButton fazLevantamento;

    public VistaConta(ModeloObservavel modelo) {
        this.modelo = modelo;
        /* B */

        saldo = new JLabel( modelo.consultaSaldo());
        fazLevantamento = new JButton("Lavantar 50 euros");

        setLayout(new FlowLayout());

        /* C */
        /* D */

        fazLevantamento.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ae) {
                    /* E */            }
        });
    }

    @Override
    public void update(Observable o, Object arg) {
        /* F */
    }
}
```

Listagem G

```
interface ICommand{
    void execute();
    void undo();
}
class ComandoFazLevantamento implements ICommand {
    private Multibanco multibanco;
    private /* A */;

    public ComandoFazLevantamento(Multibanco multibanco, /* B */) {
        this.multibanco = multibanco;
        /* C */;
    }

    @Override
    public void execute() {
        /* D */;
    }

    @Override
    public void undo() {
        // faria o deposito da mesma quantia
    }


}
```