Ariketa01

Zein izango da hurrengo programaren irteera?

```
public class Main
{
    private static int metodoa()
    {
        int balioa = 0;
        try
        {
            balioa = balioa + 1;
            balioa = balioa + Integer.parseInt("42");
            balioa = balioa + 1;
            System.out.println("Try blokearen bukaeran: " + balioa);
        }
        catch (NumberFormatException e)
        {
            balioa = balioa + Integer.parseInt("42");
            System.out.println("Catch blokearen bukaeran: " + balioa);
        }
        finally
        {
            balioa = balioa + 1;
            System.out.println("Finally blokearen bukaeran: " + balioa);
        }
        balioa = balioa + 1;
        System.out.println("Balioa itzuli aurretik: " + balioa);
        return balioa;
    }

    public static void main(String[] args)
    {
        try
        {
            System.out.println(metodoa());
        }
        catch (Exception e)
        {
            System.out.println("Salbuespena metodoa() metodoan.");
            e.printStackTrace();
        }
    }
}
```

Ariketa02

Zein izango da hurrengo programaren irteera?

```
public class Main
{
    private static int metodoa()
    {
        int balioa = 0;
        try
        {
            balioa = balioa + 1;
            balioa = balioa + Integer.parseInt("W");
            balioa = balioa + 1;
            System.out.println("Try blokearen bukaeran: " + balioa);
        }
        catch (NumberFormatException e)
        {
            balioa = balioa + Integer.parseInt("42");
            System.out.println("Catch blokearen bukaeran: " + balioa);
        }
        finally
        {
            balioa = balioa + 1;
            System.out.println("Finally blokearen bukaeran: " + balioa);
        }
        balioa = balioa + 1;
        System.out.println("Balioa itzuli aurretik: " + balioa);
        return balioa;
    }

    public static void main(String[] args)
    {
        try
        {
            System.out.println(metodoa());
        }
        catch (Exception e)
        {
            System.out.println("Salbuespena metodoa() metodoan.");
            e.printStackTrace();
        }
    }
}
```

Ariketa03

Zein izango da hurrengo programaren irteera?

```
public class Main
{
    private static int metodoa()
    {
        int balioa = 0;
        try
        {
            balioa = balioa + 1;
            balioa = balioa + Integer.parseInt("W");
            balioa = balioa + 1;
            System.out.println("Try blokearen bukaeran: " + balioa);
        }
        catch (NumberFormatException e)
        {
            balioa = balioa + Integer.parseInt("W");
            System.out.println("Catch blokearen bukaeran: " + balioa);
        }
        finally
        {
            balioa = balioa + 1;
            System.out.println("Finally blokearen bukaeran: " + balioa);
        }
        balioa = balioa + 1;
        System.out.println("Balioa itzuli aurretik: " + balioa);
        return balioa;
    }

    public static void main(String[] args)
    {
        try
        {
            System.out.println(metodoa());
        }
        catch (Exception e)
        {
            System.out.println("Salbuespena metodoa() metodoan.");
            e.printStackTrace();
        }
    }
}
```

Ariketa04

Zein izango da hurrengo programaren irteera?

```
public class Main
{
    private static int metodoa()
    {
        int balioa = 0;
        try
        {
            balioa = balioa + 1;
            balioa = balioa + Integer.parseInt("W");
            balioa = balioa + 1;
            System.out.println("Try blokearen bukaeran: " + balioa);
            throw new IOException();
        }
        catch (IOException e)
        {
            balioa = balioa + Integer.parseInt("W");
            System.out.println("Catch blokearen bukaeran: " + balioa);
        }
        finally
        {
            balioa = balioa + 1;
            System.out.println("Finally blokearen bukaeran: " + balioa);
        }
        balioa = balioa + 1;
        System.out.println("Balioa itzuli aurretik: " + balioa);
        return balioa;
    }

    public static void main(String[] args)
    {
        try
        {
            System.out.println(metodoa());
        }
        catch (Exception e)
        {
            System.out.println("Salbuespena metodoa() metodoan.");
            e.printStackTrace();
        }
    }
}
```

Ariketa05

Sortu **ZatiZero** izeneko klase bat *main* metodo batekin. Metodo honek hurrengoa egingo du:

- Bi zenbaki oso eskatu eta lehena zati bigarrenaren emaitza erakutsiko du.
- Bigarren zenbakia zero den kasuetan ematen den zati zero egoera salbuespen baten bidez kudeatuko du. "Errorea: Zati zero" mezua erakutsiko du.

Behin ondo funtzionatzen duenean, egin beharrezkoak diren aldaketak *main* metodoak salbuespenik barreiatu ez dezan.

Ariketa06

Sortu **ZenbakienBatura** izeneko klase bat *main* metodo batekin. Metodo honek zenbakiak jasoko ditu parametro bezala eta metatzen (batzen) joango da. Parametroren bat zenbakia ez bada "XXX ezin da batu" mezua erakutsiko du eta hurrengo parametroen baturarekin jarraituko du. Bukaeran honakoa erakutsiko du:

- Zenbakien batura
- Batu ez diren parametroen kopurua

Oharra: *Integer.parseInt(String)* metodoari deitzean parametroa zenbaki bat ez denean *NumberFormatException* motako salbuespena aktibatzen da.

Ariketa07

Sortu **Zatiki** izeneko klase bat. Klase honek ondorengo ezaugarriak izango ditu:

- Atributu bezala bi zenbaki oso izango ditu (zatikizuna eta zatitzailea).
- Bi eraikitzaile izango ditu. Batek bi balioak hartuko ditu parametro bezala. Besteak ez du parametrorik izango eta atributuak hasierako balio batera ezarriko ditu.
- *toString* metodoa berridatziko du bere balioak "zatikizuna / zatitzailea" formatuan itzultzeko.
- Zatikiak batu, kendu, bidertu eta zatitzeko metodoak izango ditu. Metodo hauek beste zatiki bat hartuko dute parametro bezala eta eragiketa egin ondoren beste Zatiki motako objektu bat itzuliko dute. Metodo guzti hauek eragiketaren ondoren emaitza sinplifikatzeko metodo bati deituko diote. Metodo honek Zatitzaile Komun Handiena erabiltzen du zatikizuna eta zatitzailea txikitzeko. Oharra: Kalkulatu ZKH Euklidesen algoritmoa erabilita.

Ondoren, sortu **ZatikiSalbuespena** izeneko salbuespena definitu. Salbuespen honek ondorengo ezaugarriak izango ditu:

- Existitzen den salbuespen bat hedatuko du (aukeratu egokiena).
- Eraikitzaileak karaktere-kate bat jasoko du parametro bezala eta superklasearen eraikitzaileari deituko dio karaktere-kate horrekin.

Orain, moldatu *Zatiki* klasea *ZatikiSalbuespen* bat aktibatu (*throw*) dezan errore bat ematen denean:

- Zatitzaile bezala zero zenbakia jasotzen denean.
- Zatiketa baten emaitzan zatitzaile zero duen zatiki bat lortzen denean. Kasu honetan, zatidura objektua sortu aurretik salbuespena aktibatu.

Mezua desberdina izango da salbuespen bakoitzerako. Lehen salbuespenaren kasuan "Ezin da zatitzaile bezala zero duen zatikia sortu" izango da eta bigarrenaren kasuan "Eragiketa honen emaitzak baliozkoa ez de zatiki bat sortzen du".

Bukatzeko, sortu **Main** klase bat *main* metodo batekin. Metodo honek *Zatiki* klasea frogatuko du salbuespenak jaso eta dagokien mezuak erakutsiz.

Ariketa08

Moldatu 6. unitateko 1. ariketa multzoko 3. ariketa (zinemarena). Zehazki, eraikitzaileak moldatu sarrerako balioak baliozkoak ez direnean salbuespen pertsonalizatuak aktiba ditzaten.

- Pelikula klasean: izenburu huts bat edo iraupen edo urte desegoki bat sartzen denean.
- Zinema klasean: izen huts bat edo areto kopuru desegokia sartzen denean.

Salbuespen hauek probatzeko egin *main* metodoan beharrezkoak diren aldaketak.