

Hands on Practice – Objects

Exercise 1 – Stable Programming

We are going to program a stable with cattle. We start with a program based on the one in the lecture notes, illustrating objects with references. Download the files `Animal.java`, `Trough.java`, and `Stable.java`, or type in the class definitions below. You can put the classes in one file if you like, but having them in separate files is usually more convenient.

```
class Animal {  
  
    String name = "noname";  
  
    int stomach; // number of meals  
  
    Trough trough;  
  
    // constructor  
  
    Animal(String n) {  
  
        name = n;  
  
        stomach = 0;  
  
    }  
  
    void setTrough(Trough t) {  
  
        trough = t;  
  
    }  
  
    void eat() {  
  
        trough.give();  
  
        stomach = stomach + 1;  
  
    }  
}
```

```
}
```

```
}
```

```
class Trough {
```

```
int contents; //number of meals
```

```
int sn; //serial number
```

```
Trough(int n) {
```

```
sn = n;
```

```
}
```

```
void fill(int m) {
```

```
contents = contents+m;
```

```
}
```

```
void give() {
```

```
contents = contents-1;
```

```
}
```

```
}
```

```
class Stable {
```

```
Animal animal1;
```

```
Trough trough1;
```

```
void farm() {
```

```
animal1 = new Animal("Klaartje");
```

```
trough1 = new Trough(1);
```

```

animal1.setTrough(trough1);

trough1.fill(4);

animal1.eat();

animal1.eat();

}

public static void main(String[] args) {

    Stable s = new Stable();

    s.farm();

}

}

```

1. As you see, we have removed the print statements in the methods. Instead, we want to have separate print methods that print a status report of the called object. Add a method void print() to the class Animal that prints a report like this:

Cow Klaartje has 2 meals in her stomach. Use it in the farm method.

```

package com.example.hello;
import java.util.*;
import java.lang.*;
//vineet codes java
// sout-- printing single lines
// psvm --> main file

import java.util.Scanner;

class Animal {

```

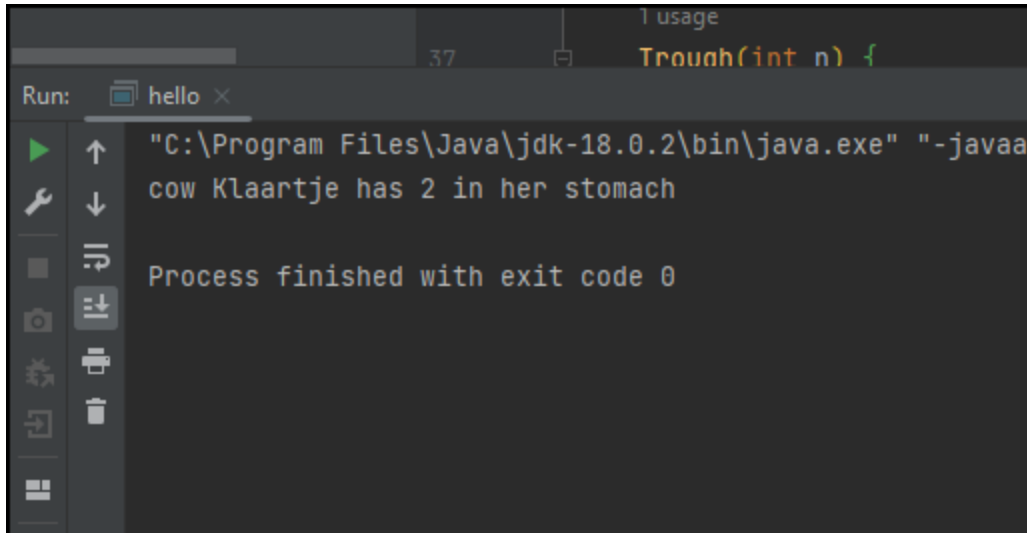
```

String name = "noname";
int stomach; // number of meals
Trough trough;
// constructor
Animal(String n) {
    name = n;
    stomach = 0;
}
void setTrough(Trough t) {
    trough = t;
}
void eat() {
    trough.give();
    stomach = stomach + 1;
}

void print()
{
    System.out.println("cow " + name + " has " + stomach + " in her
stomach ");
}
}
class Trough {
    int contents; //number of meals
    int sn; //serial number
    Trough(int n) {
        sn = n;
    }
    void fill(int m) {
        contents = contents+m;
    }
    void give() {
        contents = contents-1;
    }
}
class Stable {
    Animal animall;
    Trough trough1;

    void farm() {
        animall = new Animal("Klaartje");
        trough1 = new Trough(1);
        animall.setTrough(trough1);
        trough1.fill(4);
        animall.eat();
        animall.eat();
        animall.print();
    }
}
public class hello {
    public static void main(String[] args) {
        Stable s = new Stable();
        s.farm();
    }
}

```



```
Run: hello x
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaa
cow Klaartje has 2 in her stomach
Process finished with exit code 0
```

2. Add a similar print method to Trough and Stable. The latter will have to call the print methods of the Animal and Trough objects. Use the new methods.

```
package com.example.hello;
import java.util.*;
import java.lang.*;
//vineet codes java
// sout-- printing single lines
// psvm --> main file

import java.util.Scanner;

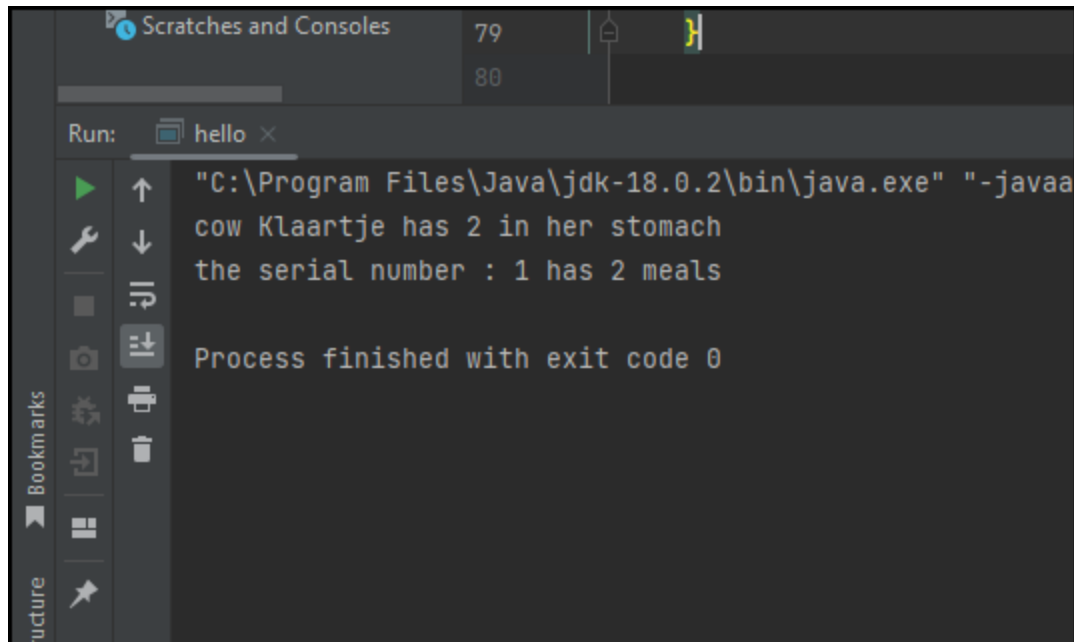
class Animal {
    String name = "noname";
    int stomach; // number of meals
    Trough trough;
    // constructor
    Animal(String n) {
        name = n;
        stomach = 0;
    }
    void setTrough(Trough t) {
        trough = t;
    }
    void eat() {
        trough.give();
        stomach = stomach + 1;
    }
    void print() {
        System.out.println("cow " + name + " has " + stomach + " in her
```

```

stomach ");
    }
}
class Trough {
    int contents; //number of meals
    int sn; //serial number
    Trough(int n) {
        sn = n;
    }
    void fill(int m) {
        contents = contents+m;
    }
    void give() {
        contents = contents-1;
    }

    void print()
    {
        System.out.println("the serial number : " + sn + " has " + contents + "
meals ");
    }
}
class Stable {
    Animal animal1;
    Trough trough1;
    void farm() {
        animal1 = new Animal("Klaartje");
        trough1 = new Trough(1);
        animal1.setTrough(trough1);
        trough1.fill(4);
        animal1.eat();
        animal1.eat();
        animal1.print();
        trough1.print();
    }
}
public class hello {
    public static void main(String[] args) {
        Stable s = new Stable();
        s.farm();
    }
}

```



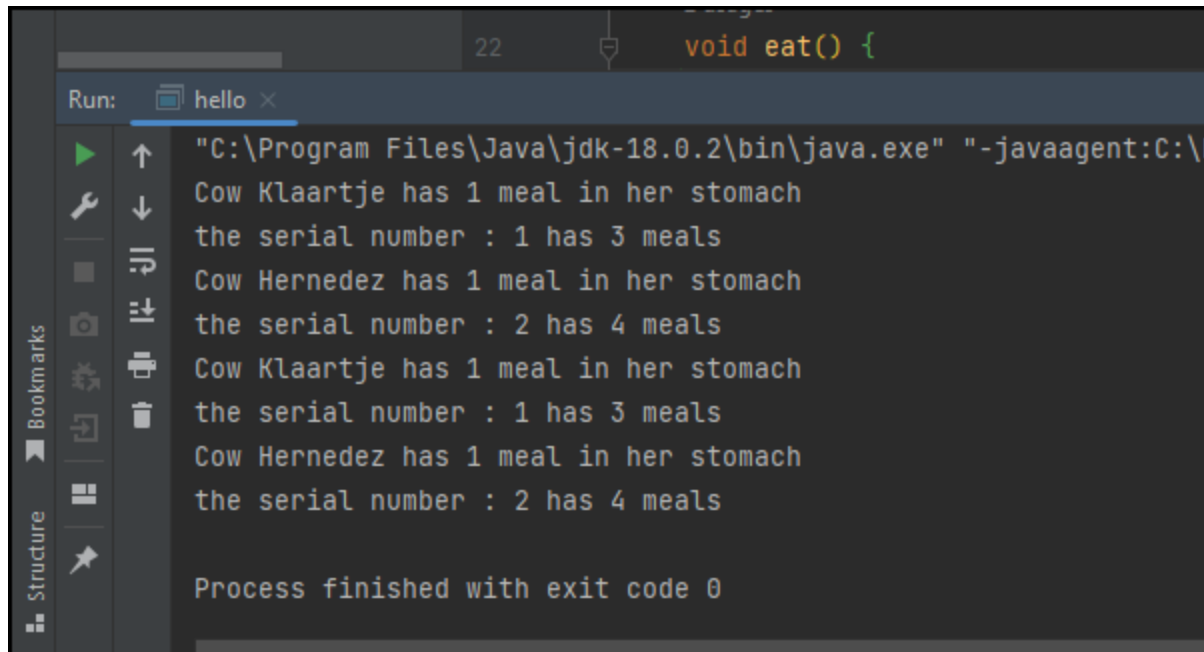
3. Now add another Animal and another Trough object to the Stable. Associate the new Trough to the new Animal. Let them eat and print reports in several stages of the process.

```
package com.example.hello;
import java.util.*;
import java.lang.*;
//vineet codes java
// sout-- printing single lines
// psvm --> main file
import java.util.Scanner;
class Animal {
    String name = "noname";
    int stomach; // number of meals
    Trough trough;
    // constructor
    Animal(String n) {
        name = n;
        stomach = 0;
    }
    void setTrough(Trough t) {
        trough = t;
    }
}
```

```

    void eat() {
        trough.give();
        stomach = stomach + 1;
    }
    void print()
    {
        System.out.println("Cow " + name + " has " + stomach + " meal in her
stomach ");
    }
}
class Trough {
    int contents; //number of meals
    int sn; //serial number
    Trough(int n) {
        sn = n;
    }
    void fill(int m) {
        contents = contents+m;
    }
    void give() {
        contents = contents-1;
    }
    void print()
    {
        System.out.println("the serial number : " + sn + " has " + contents + "
meals ");
    }
}
class Stable {
    Animal animal1 , animal2;
    Trough trough1 , trough2;
    void farm() {
        animal1 = new Animal("Klaartje");
        animal2 = new Animal("Hernandez");
        trough1 = new Trough(1);
        trough2 = new Trough(2);
        animal1.setTrough(trough1);
        animal2.setTrough(trough2);
        trough1.fill(4);
        trough2.fill(5);
        animal1.eat();
        animal2.eat();
        animal1.print();
        trough1.print();
        animal2.print();
        trough2.print();
    }
}
public class hello {
    public static void main(String[] args) {
        Stable s = new Stable();
        s.farm();
        Stable s1 = new Stable();
        s1.farm();
    }
}

```

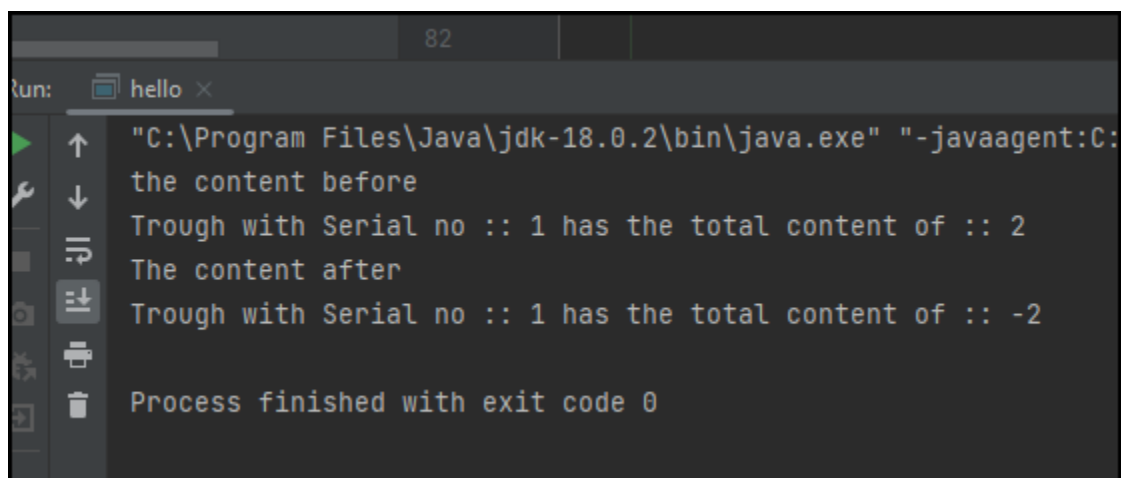



```
Run: hello x
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\f
Cow Klaartje has 1 meal in her stomach
the serial number : 1 has 3 meals
Cow Hernandez has 1 meal in her stomach
the serial number : 2 has 4 meals
Cow Klaartje has 1 meal in her stomach
the serial number : 1 has 3 meals
Cow Hernandez has 1 meal in her stomach
the serial number : 2 has 4 meals

Process finished with exit code 0
```

4. When a trough is empty an animal can still eat from it. The contents of the trough will get a negative value in that case. This is not very realistic. Therefore, change the eat method such that it checks whether the trough has enough food before it calls the method give.

Add a method to Trough that reports the contents of the trough and use this method in eat .



```
Run: hello x
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:
the content before
Trough with Serial no :: 1 has the total content of :: 2
The content after
Trough with Serial no :: 1 has the total content of :: -2

Process finished with exit code 0
```

```

package com.example.hello;
import java.util.*;
import java.lang.*;
//vineet codes java
// sout-- printing single lines
// psvm --> main file
import java.util.Scanner;
class Animal {
    String name = "noname";
    int stomach; // number of meals
    Trough trough;
    // constructor
    Animal(String n) {
        name = n;
        stomach = 0;
    }
    void setTrough(Trough t) {
        trough = t;
    }
    void eat() {
        if(trough.contents <= 0) {
            System.out.println("The trough is empty!!! fill more food ");
        }
        else {
            trough.give();
            stomach = stomach + 1;
        }
    }
    void print()
    {
        System.out.println("Cow " + name + " has " + stomach + " meal in her
stomach ");
    }
}
class Trough {
    int contents; //number of meals
    int sn; //serial number
    Trough(int n) {
        sn = n;
    }
    void fill(int m) {
        contents = contents+m;
    }
    void give() {
        contents = contents-1;
    }

    void print()
    {
        System.out.println("the serial number : " + sn + " has " + contents + "
meals ");
    }
}

```

```

        void contentinfo()
        {
            System.out.println("Trough with Serial no :: " + sn + " has the total
content of :: " + contents );
        }
    }
}

class Stable {
    Animal animal1 ;
    Trough trough1 ;
    void farm() {
        animal1 = new Animal("Klaartje");
        trough1 = new Trough(1);
        animal1.setTrough(trough1);
        trough1.fill(2);
        System.out.println("the content before");
        trough1.contentinfo();
        animal1.eat();
        animal1.eat();
        animal1.eat();
        animal1.eat();
        System.out.println("The content after");
        trough1.contentinfo();
        // animal1.print();
        //trough1.print();
    }
}

public class hello {
    public static void main(String[] args) {
        Stable s = new Stable();
        s.farm();
    }
}

```

```

Run: hello x
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Com
the content before
Trough with Serial no :: 1 has the total content of :: 2
Eating now!!
Eating now!!
The trough is empty!!! fill more food
The trough is empty!!! fill more food
The content after
Trough with Serial no :: 1 has the total content of :: 0

Process finished with exit code 0

```

5. Add a third Animal to the Stable and let this one share her Trough with one of the others.

Again, fill, eat, and report liberally. Observe that indeed one of the troughs will empty faster than the other one.

```
package com.example.hello;
import java.util.*;
import java.lang.*;
//vineet codes java
// sout-- printing single lines
// psvm --> main file

import java.util.Scanner;

class Animal {
    String name = "noname";
    int stomach; // number of meals
    Trough trough;
    // constructor
    Animal(String n) {
        name = n;
        stomach = 0;
    }
    void setTrough(Trough t) {
        trough = t;
    }
    void eat() {
        if(trough.contents <= 0) {
            System.out.println("The trough is empty!!! fill more food ");
        }
        else {System.out.println("the animal , whose name is ::" + name + "
Eating now!!");
            trough.give();
            stomach = stomach + 1;
        }
    }

    void print()
    {
        System.out.println("Cow " + name + " has " + stomach + " meal in her
stomach ");
    }
}

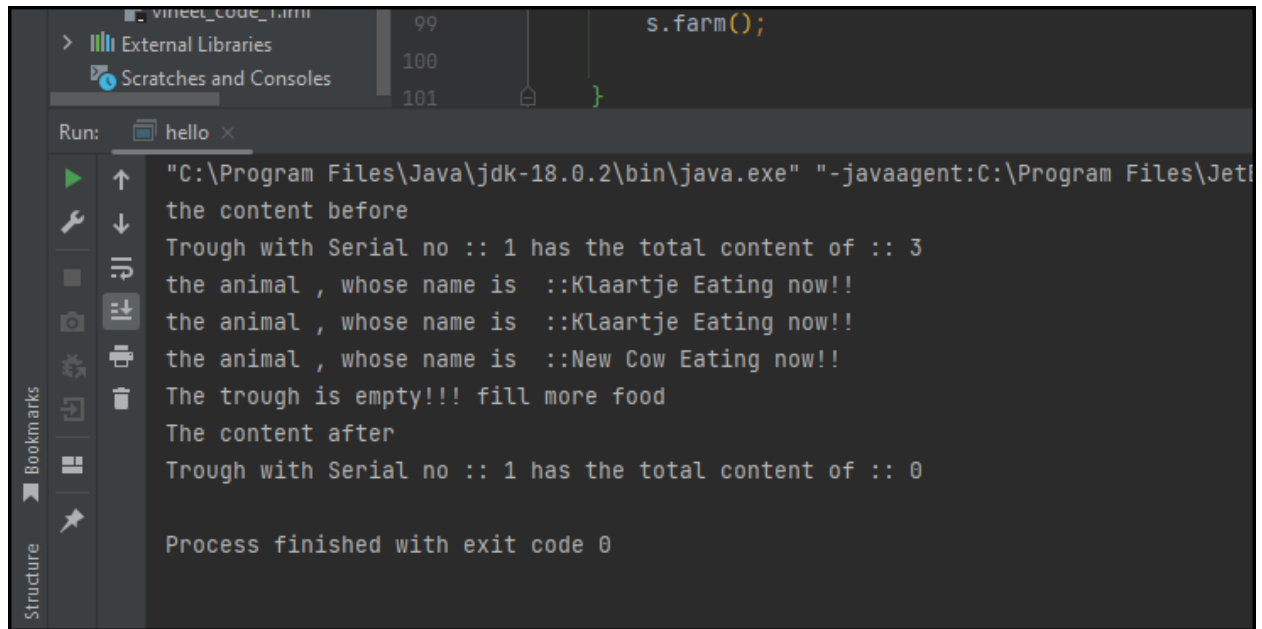
class Trough {
```

```

    int contents; //number of meals
    int sn; //serial number
    Trough(int n) {
        sn = n;
    }
    void fill(int m) {
        contents = contents+m;
    }
    void give() {
        contents = contents-1;
    }
    void print()
    {
        System.out.println("the serial number : " + sn + " has " + contents + "
meals ");
    }
    void contentinfo()
    {
        System.out.println("Trough with Serial no :: " + sn + " has the total
content of :: " + contents );
    }
}
class Stable {
    Animal animal1 , animal2 ;
    Trough trough1 ;
    void farm() {
        animal1 = new Animal("Klaartje");
        trough1 = new Trough(1);
        animal2 = new Animal("New Cow") ;
        animal1.setTrough(trough1);
        animal2.setTrough(trough1);
        trough1.fill(3);

        System.out.println("the content before");
        trough1.contentinfo();
        animal1.eat();
        animal1.eat();
        animal2.eat();
        animal1.eat();
        System.out.println("The content after");
        trough1.contentinfo();
    }
}
public class hello {
    public static void main(String[] args) {
        Stable s = new Stable();
        s.farm();
    }
}

```



```
99 s.farm();
100
101 }

Run: hello x
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Program Files\Jeti
the content before
Trough with Serial no :: 1 has the total content of :: 3
the animal , whose name is ::Klaartje Eating now!!
the animal , whose name is ::Klaartje Eating now!!
the animal , whose name is ::New Cow Eating now!!
The trough is empty!!! fill more food
The content after
Trough with Serial no :: 1 has the total content of :: 0

Process finished with exit code 0
```

Exercise 2 – Using objects

In this exercise, you will use a class written by someone else. It is the class `Coin` that models a coin for tossing (coin flipping). Unfortunately, the coin may be biased, i.e., the chance of heads or tails deviates significantly from 50%. It is your task to find out whether the coin is biased or unbiased.

The class has a constructor that creates a coin with a random bias. It has the following methods.

- `int toss()`

This method tosses the coin and returns the result as an `int`: 0 or 1.

- `void guessBiased(boolean)`

This method expresses the guess that the coin is biased (parameter `true`), or unbiased (parameter `false`). For each coin that you create, the method can only be called once, to prevent that you

from keeping guessing.

1. Try out the Coin class with the following program (also available on the web).

```
class CoinTester {  
  
    void test() {  
  
        Coin coin = new Coin();  
  
        int result;  
  
        result = coin.toss();  
  
        System.out.println("toss: "+ result);  
  
        coin.guessBiased(true);  
  
    }  
  
    public static void main(String[] args) {  
  
        CoinTester ct = new CoinTester();  
  
        ct.test();  
  
    }  
}
```

3. Now write your own test program. Two approaches are possible:

(a) Write a loop that tosses the coin several times and counts the number of 0s and 1s. After a reasonable number of tosses, the program decides whether the coin is biased or not, based on the recorded numbers.

(b) Write a loop that repeatedly asks the user whether she wants to toss and if so, tosses the coin and prints the result. If instead the user wants to stop, the program asks for the guess and expresses the guess to the coin.

```
package com.example.hello;
import java.util.*;
import java.lang.*;
//vineet codes java
// sout-- printing single lines
// psvm --> main file

import java.util.Scanner;

class Coin
{
    public final int HEADS = 0;
    public final int TAILS = 1;
    private int face;
    // Sets up the coin by flipping it initially.
    public Coin ()
    {
        flip();
    }
    // Flips the coin by randomly choosing a face.
    public void flip ()
    {
        face = (int) (Math.random() * 2);
    }
    public int getFace ()
    {
        return face;
    }
    // Returns the current face of the coin as a string.
    public String toString()
    {
        String faceName;
        if (face == HEADS)
            faceName = "Heads";
        else
            faceName = "Tails";
        return faceName;
    }
}

class CountFlips
{
    // Flips a coin multiple times and counts the number of heads and tails
    that result.
    public static void main (String[] args)
    {
        final int NUM_FLIPS = 1000;
```



```

        int heads = 0, tails = 0;
        Coin myCoin = new Coin(); // instantiate the Coin object
        for (int count=1; count <= NUM_FLIPS; count++)
        {
            myCoin.flip();
            if (myCoin.getFace() == myCoin.HEADS)
                heads++;
            else
                tails++;
        }
        System.out.println ("The number flips: " + NUM_FLIPS);
        System.out.println ("The number of heads: " + heads);
        System.out.println ("The number of tails: " + tails);
    }
}

```

```

public class hello {

```

```

    public static void main(String[] args) {

```

```

        final int NUM_FLIPS = 1000;
        int heads = 0, tails = 0;
        Coin myCoin = new Coin(); // instantiate the Coin object
        for (int count=1; count <= NUM_FLIPS; count++)
        {
            myCoin.flip();
            if (myCoin.getFace() == myCoin.HEADS)
                heads++;
            else
                tails++;
        }
        System.out.println ("The number flips: " + NUM_FLIPS);
        System.out.println ("The number of heads: " + heads);
        System.out.println ("The number of tails: " + tails);

```

```

        float estimator = (float) heads/ (float) (heads + tails ) ;

```

```

        double s = 0.5*Math.pow(0.5 , NUM_FLIPS) ;

```

```

        if(estimator < 0.5)
        {
            System.out.println("Fair coin!! with estimator value::" + estimator);

```

```

        }else {
            System.out.println("COin is fake !! becoz the estimator value :: " +
            estimator);

```

```

        }
        //System.out.println("estimator ::" + estimator);

```

```

    }

```

```

}

```

