Online2

Monsters come and go from the zoo over the years. Monsters each have a name, a number of legs, a number of eyes and they may fly (or not). Monsters also have a favorite food. Their food has a name and a typical serving size measured in kilograms. The zoo's keepers need to keep track of how much each monster has been fed during its lifetime. Sometimes monsters get in fights and they might lose an eye or a limb. This information needs to be maintained in the zoo's database. The zoo-keeper can add monsters by only mentioning their name or with all attributes.

We provide two classes **`TestZoo`** and **`TestMonster`**. Now write the source codes of class **`Zoo`**, class **`Monster`** and class **`Food`** so that the given classes work.

Implementing class `Food`

This class defines methods to get the name of a food, find the serving size of the food. However, the skeleton that we have provided is missing the implementation of these methods. Fill in the body of the methods with appropriate code.

Notes:

- While implementing constructor method, remember each food has a name and a typical serving size measured in kilograms.
- Serving size can never be negative

Implementing class `Monster`

You will need to define and implement the missing methods and member variables of class Monster. Read the main method of class TestMonster and investigate the compile-time errors to figure out what methods are missing.

Notes:

- While implementing constructor method, remember the zoo-keeper can add monsters by only mentioning their name or with all attributes (with monter's name, number of legs, number of eyes, they may fly or not, favorite food and the number of times it has been fed during its lifetime)
- Number of limbs or eyes can never be negative
- You will find the eat() method to have two versions

Implementing class `Zoo`

This class defines methods to add a single monster or an array of monsters, get the number of monsters to the zoo. However, the skeleton that we provide is missing the implementations of the methods. Fill in the body of the methods with appropriate code.

Notes:

- Number of monsters cannot surpass the capacity of zoo

Submission

Write the 3 classes in a single .java file, rename the file to your 10 digit student ID and then submit.

Classes

```
class Food {
    private String name;
    private double servingSize;

    //creates a new food
    public Food(String name, double servingSize) {
        //implement this method
    }

    //gets the name of the food
    public String getName() {
        //implement this method
    }

    //gets the serving size of the food
    public double getServingSize() {
        //implement this method
    }
}

//implement this class
class Monster {

}


class Zoo {
    private int capacity;
    private int numberOfMonsters;
    private Monster[] entriedMonsters;

    //creates a new zoo
    public Zoo(int capacity) {
        //implement this method
    }

    //adds an array of monsters to the zoo
    public void addAMonster(Monster[] monsters){
        //implement this method
    }

    //adds a monster to the zoo
    public void addAMonster(Monster monster){
        //implement this method
    }

    //returns the number of monsters in zoo
    public int getNumberOfMonsters() {
        //implement this method
    }
```

```
    // returns the name of the monsters that have been fed x times in their
    lifetime
    public static String getMonsters(int numTimesFed){

    }

    // returns the name of the monsters whose favorite food is food
    public static String getMonsters(Food food){

    }

}


public class TestMonster {

    public static void main(String[] args) {

        Food cookies = new Food("cookie", 3.0);
        System.out.println(cookies.getName());
        System.out.println(cookies.getServingSize());

        Monster cookieMonster = new Monster("CookieMonster", 4, 2, false, cookies,
                                            0);
        cookieMonster.looseALimb();
        cookieMonster.looseALimb();
        cookieMonster.looseAnEye();
        System.out.println(cookieMonster.getNumberOfLimbs());


        Food notCookies = new Food("broccoli", 1.0);
        Food meat[]=new Food[2];
        meat[0]=new Food("Chicken Meat", 1.0);
        meat[1]=new Food("Beef", 1.0);

        cookieMonster.eat(cookies);
        System.out.println(cookieMonster.getLifetimeFeed());

        cookieMonster.eat(meat);
        System.out.println(cookieMonster.getLifetimeFeed());

        cookieMonster.eat(cookies);
        System.out.println(cookieMonster.getLifetimeFeed());


        cookieMonster.eat(notCookies);
        System.out.println(cookieMonster.getLifetimeFeed());


        System.out.println(cookieMonster);

    }

}
```

```
public class TestZoo {
    public static void main(String[] args) {
        Zoo theZoo = new Zoo(5);// 5 refers to the capacity

        // add some monsters
        Monster monsters[] = new Monster[4];
        monsters[0] = new Monster("m1", 4, 2, true, new Food("cookies"), 5);
        monsters[1] = new Monster("m2", 7, 1, false, new Food("spinach"), 0);
        monsters[2] = new Monster("m3");
        monsters[3] = new Monster("m4", 0, 6, false, new Food("humans"), 1);
        Monster m4 = new Monster("m5");

        theZoo.addAMonster(monsters);
        theZoo.addAMonster(m4);
        System.out.println(theZoo.getNumberOfMonsters() + " monsters");


        //add another monster
        Food cookies = new Food("Cookies", 3.0);
        Monster badMonster = new Monster("Bad Monster", 4, 2, false, cookies,0);

        theZoo.addAMonster(badMonster);
        System.out.println(theZoo.getNumberOfMonsters() + " monsters");

        System.out.println(Zoo.getMonsters(new Food("humans")));
        System.out.println(Zoo.getMonsters(0));


    }

}
```

Output

Output of TestMonster
cookie
3.0
2
1
3
4
5
cookie has 2 limbs and 1 eyes.

Output of TestZoo
5 monsters
Zoo is full
5 monsters
m4
m2, m3, m5