

CSC 131 SP'19

In-Class Activity

Classes and UML

The String class provides methods for working with text. The Random class provides methods for generating random numbers. In this activity, you'll learn how to make your own classes that represent everyday objects.

Content Learning Objectives

After completing this activity, students should be able to:

- Define the terms: attribute, method, constructor, instance.
- Implement non-static methods based on a UML diagram.
- Implement class inheritance based on a UML diagram.

Process Skill Goals

During the activity, students should make progress toward:

- Writing method signatures exactly as shown in a UML diagram. (Information Processing)



This activity is modified based on the work of © 2017 Chris Mayfield and Helen Hu, which is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Model 1 The Die Class

When you define a class in Java, you are defining a new data type. Classes have *attributes* (data) and *methods* (code). A *class diagram* is a graphical summary of the attributes and methods.

Simulates a Die object.

```
public class Die { private
    int face;
```

Constructs a new die with a random face value.

```
    public Die() { this.face
        = 1;
    }
```

Gets the current face value of the die.

©return current face value of the die

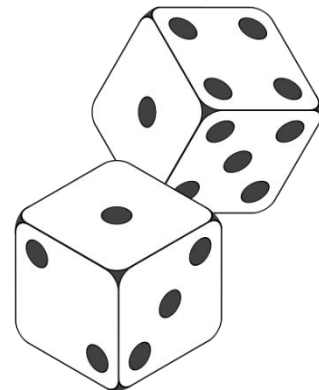
```
    public int getFace() { return
        this.face;
    }
```

Simulates the roll of the die.

©return new face value of the die

```
    public int roll() {
        this.face = (int) (Math.random() 6) + return 1;
        this.face;
    }
}
```

Die
- face: int
+Die() +getFace(): int +roll(): int



Questions (15 min)

Start time: _____

1. What are the attributes of Die? What are the methods?

Attributes: face

Methods: Die, getFace, and roll

2. In the class diagram, what do the -and +symbols represent? What does the :represent?

Plus means public

Minus means private

Colon refers to data type

3. Write a statement that *declares* a Dievariable named lucky.

Die lucky;

4. Each *instance* of a class (in memory) is called an object. Write a statement that *instantiates* a **new** Die object and assigns it to lucky.

lucky = new Die();

5. When you instantiate an object, you invoke a *constructor*. This method has no return type and has the same name as the class itself. What does the Dieconstructor do?

It initializes the face attribute to 1.

6. Notice how the roll method refers to face, yet that variable is not declared in the method. What does the rollmethod change, in terms of the Dieobject?

It updates the value of the face attribute.

7. What is the purpose of the getFace method? Show how you would use it in a main method of another class.

In a main method, you would do something like: `System.out.println(lucky.getFace());`

Model 2 The Circle Class

Unified Modeling Language (UML) provides a way of graphically illustrating a class's design, independent of the programming language.

Circle
-radius: double
+Circle(radius:double) +getRadius(): double +setRadius(radius:double) +area(): double +circumference(): double <u>+main(args:String[])</u>

Questions (18 min)

Start time: _____

8. What are the attributes and methods of Circle, and what is their *visibility*?
The attribute radius is private, and the methods Circle, area, circumference, getRadius, and setRadius are public.
9. Based on Model 1 and Model 2, what is typically **public** and what is typically **private**?
Attributes are typically private, and methods are typically public.
10. How would you declare a variable named unit that is a Circle object? How would you instantiate a circle with a radius of 1.0 and assign it to unit?
Circle unit;
unit = new Circle(1.0);
11. Write the code inside Circle.java (you need to create this class in Eclipse) that declares the radius attribute.
private double radius;
12. Write the code for getRadius. (Don't worry about Javadoc comments for this activity.)
public double getRadius() {
 return this.radius;
}

13. Write the code for `setRadius`. Note there are two variables named `radius`: the parameter of `setRadius`, and `this.radius` for the object itself. Before you set the radius, first check if the parameter is negative, and if it is, set `this.radius` to zero instead.

```
public void setRadius(double radius) {  
    if (radius >= 0) {  
        this.radius = radius;  
    } else {  
        this.radius = 0;  
    }  
}
```

14. Write the complete code for `area` and `circumference`. The area of a circle is πr^2 , and the circumference is $2\pi r$. Ideally, each method should be one line of code. (In Java, you may use the final constant `Math.PI` for your convenience.)

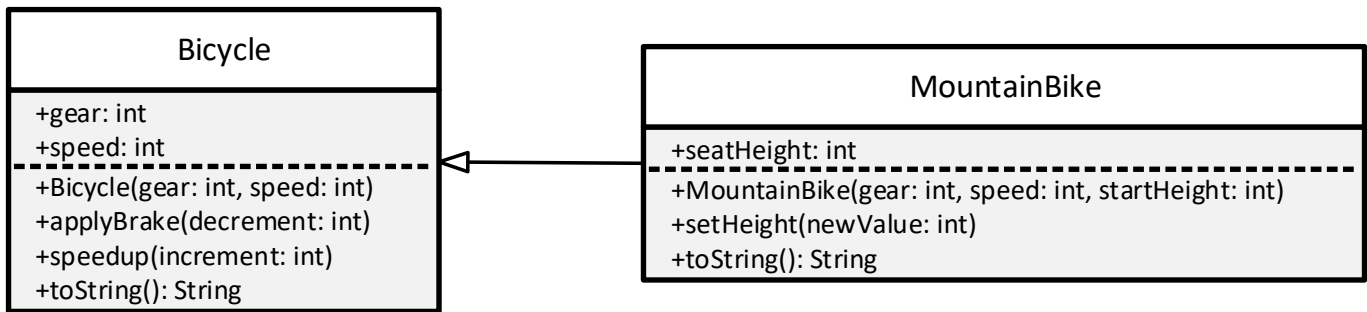
```
public double area() {  
    return Math.PI * radius * radius;  
}  
  
public double circumference() {  
    return 2.0 * Math.PI * radius;  
}
```

15. Write a main method that creates a `Circle` object with a radius of 2.0 and displays its area and circumference on the screen.

```
public static void main(String[] args) {  
    Circle circle = new Circle(2.0);  
    System.out.println("Circle area = " + circle.area());  
    System.out.println("Circle circ = " + circle.circumference());  
}
```

Model 3 The Bicycle & MountainBike Classes

Unified Modeling Language also provides a way to represent inheritance between classes.



Questions (15 min)

Start time: _____

16. What does represent? (i.e., what is the name of this relationship?)

Parent and Child class

17. In terms of Object-Oriented Programming, which one between Bicycle and MountainBike is the super class?

Bicycle

For question 18, 19, 20, please write the code in Eclipse.

18. A stub is a function that has the expected signature (i.e. name and accepted arguments). Try to create MountainBike with method stubs.

19. For each MountainBike object, its initial seatHeight is startHeight. Complete the constructor.

20. For each MountainBike object, method setHeight() can update seatHeight to newValue. Complete method setHeight().

21. For MountainBike class, its toString() is implemented as follows:

```
public String toString(){
    return (super.toString()+
        "\nseat height is "+seatHeight);
}
```

What's the output when the following test class is executed?

```
public class Test{
    public static void main(String args[]){
        MountainBike mb = new MountainBike(3, 100, 25);
        System.out.println(mb.toString());
    }
}
```

Gear is 3, Speed is 100, and Height is 25.