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

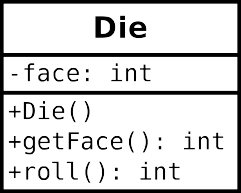
*During the activity, students should make progress toward:*

* 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.

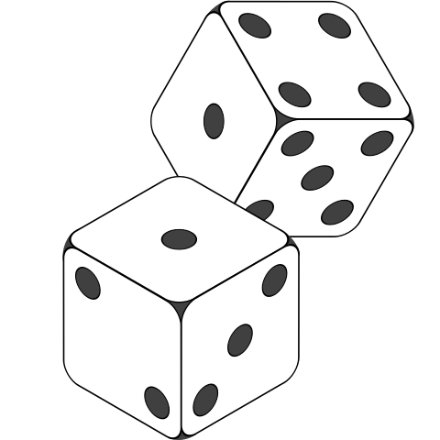
©return current face value of the die

public int getFace() { return this.face;

}

Simulates the roll of the die.

©return new face value of the die



1;

public int roll() {

this.face = (int) (Math.random() 6) + return this.face;

}

}

## Questions (15 min) Start time:

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

*It has six faces. Each of the six faces is a value between 1-6. However, in code it is from 0-5 but it adds 1. There is a math random function that chooses a value then displays it.*

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

*It represents the attribute as to whether the it is a private or public . “:” declare the value as and object.*

1. Write a statement that *declares* a Die variable named lucky.

*Lucky = new Die();*

1. 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.

Die Lucky = new Die();

1. 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 Die constructor do?

*It’s applies all of the attributes to the object you have created.*

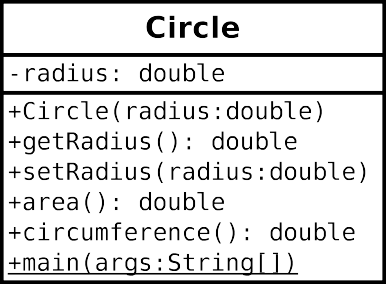
1. Notice how the roll method refers to face, yet that variable is not declared in the method. What does the roll method change, in terms of the Die object?

The roll method changes the face of the die object

1. What is the purpose of the getFace method? Show how you would use it in a main method of another class.  
   The getFace method is an accessor method to to retrieve a value from an object.

# Model 2 The Circle Class

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



## Questions (18 min) Start time:

1. What are the attributes and methods of Circle, and what is their *visibility*?

The Attribute is radius

The methods are circle(),getRadius();,setRadius();,area();,circumference(); and the main method

1. Based on [Model 1](#_bookmark0) and [Model 2,](#_bookmark1) what is typically public and what is typically private?

Nomallly, it looks like methods and constructors are marked as public while private values are uncontrolled variable like in the example being the radius. This value is not a constant. So I guess you could say public variables are constant and private variables are normally not constant.

1. 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 = new Circle (1.0);

1. Write the code inside Circle.java (you need to create this class in Eclipse) that declares the radius attribute.

Private double radius;

1. Write the code for getRadius. (Don’t worry about Javadoc comments for this activity.)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 double getRadius() {

return radius;

}

1. Write the complete code for area and circumference. The area of a circle is *πr*2, and the circumference is 2*πr*. 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 \* Math.PI \* radius;

}

1. 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("Area: " + circle.area());

System.out.println("Circumference: " + circle.circumference());

}

Circle Code:

public class Circle {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

if (radius < 0) {

this.radius = 0;

} else {

this.radius = radius;

}

}

public double area() {

return Math.PI \* radius \* radius;

}

public double circumference() {

return 2 \* Math.PI \* radius;

}

public static void main(String[] args) {

Circle circle = new Circle(2.0);

System.out.println("Area: " + circle.area());

System.out.println("Circumference: " + 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:

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

Parent child class so mountain bike is based. It is a inheritance function.

So bicycle would be superclass and mountain bike would be subclass

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

Bicicycle is superclass.

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

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

Public class MountainBike extends Bicycle{

Private int seatHeight;

Private int startHeight;

}

Public MoutainBike(int gear, int speed, int startHeight){

}

Public void setHeight(int newValue){

}

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

public class MountainBike extends Bicycle{

private int seatHeight;

private int startHeight;

}

public MoutainBike(int gear, int speed, int startHeight){

Super(gear, speed);

this.seatHeight = startHeight;

this.startHeight = startHeight;

}

public void setHeight(int newValue){

This.seatHeight = newValue; // sets the value as the new height

}

public String toString(){

return (super.toString()+ "\nseat height is "+seatHeight);

}

20. For each MountainBike object, method setHeight() can update seatHeight to newValue. Complete method setHeight(). Completed that in the box above^^^^

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());

    }

}

The output return would be since their isn’t really any text

3 100 seat height is 25”

MountainBike Code:

// Bicycle.java (Superclass)

public class Bicycle {

private int gear;

private int speed;

// Constructor

public Bicycle(int gear, int speed) {

this.gear = gear;

this.speed = speed;

}

// Default toString for Bicycle

public String toString() {

return "Gear: " + gear + ", Speed: " + speed;

}

}

// MountainBike.java (Subclass)

public class MountainBike extends Bicycle {

private int seatHeight;

private int startHeight;

// Constructor

public MountainBike(int gear, int speed, int startHeight) {

super(gear, speed); // Call the Bicycle constructor

this.seatHeight = startHeight;

this.startHeight = startHeight;

}

// Method to update seatHeight

public void setHeight(int newValue) {

this.seatHeight = newValue;

}

// toString method as specified

public String toString() {

return (super.toString() +

"\nseat height is " + seatHeight);

}

}

// Test.java (Test class)

public class Test {

public static void main(String args[]) {

MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}

}