CS 113 – Computer Science I

Lecture 12 – Objects

Adam Poliak
10/25/2022
Announcements

• Assignment 06
  • Due Thursday 10/27

• Sharing code

• Mid-semester feedback
Object-oriented programming (OOP)

Method for designing programs in terms of objects

Recall: Top-down design

• the “nouns” in your feature list correspond to classes/data

• the “verbs” correspond to methods
Using objects: some special methods

The **constructor method** is called when you do a `new`

**accesors (aka getters)**
- return the values of instance variables

**mutators (aka setters)**
- set the values of instance variables

**toString()**
- returns a string representation of an object
Defining classes

By defining our own classes, we can create our own data types

A class definition contains

- the data contained by the new type (instance variables)

- the operations supported by the new type (instance methods)
Example: Defining a class `Point`

What data should it have?
- X-coordinate
- y-coordinate
- Name
- color

What operations should it support?
this

`this` is a special keyword that refers to the object inside an instance method

Analogy:
Visualizing programs with objects

```java
class Point {
    public double x = -1.0;
    public double y = -1.0;

    public Point() {
        this.x = 0;
        this.y = 0;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void add(Point p) {
        this.x = this.x + p.getX();
        this.y = this.y + p.getY();
    }

    public static void main(String[] args) {
        Point p = new Point();
        Point p2 = new Point(3, 5);

        p.add(p2);
    }
}
```
Draw a stack diagram
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

public static void main(String[] args) {
    Point p = new Point();
    Point p2 = new Point(-4, 3);
    p.add(p2);
}
Draw a stack diagram

Function Stack:

Created objects
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

public static void main(String[] args) {
    Point p = new Point();
    Point p2 = new Point(-4, 3);
    p.add(p2);
}
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

public static void main(String[] args) {
    Point p = new Point();
    Point p2 = new Point(-4, 3);
    p.add(p2);
}
Draw a stack diagram

**Function Stack:**

**Created objects**

**Main:**
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
    public static void main(String[] args) {
        Point p = new Point();
        Point p2 = new Point(-4, 3);
        p.add(p2);
    }
}
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

public static void main(String[] args) {
    Point p = new Point();
    Point p2 = new Point(-4, 3);
    p.add(p2);
}
Draw a stack diagram

Function Stack:
Point():

Created objects

Main:
Draw a stack diagram

Function Stack:

Point():
  • this

Created objects

Main:
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }

    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }

    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }

    public static void main(String[] args) {
        Point p = new Point();
        Point p2 = new Point(-4, 3);
        p.add(p2);
    }
}
Draw a stack diagram

**Function Stack:**

Point():
  - this

**Created objects**

- x: 0
- y: 0

**Main:**
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;
    
    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
    
    public static void main(String[] args) {
        Point p = new Point();
        Point p2 = new Point(-4, 3);
        p.add(p2);
    }
}
Draw a stack diagram

Function Stack:
- Point():
  - this

Created objects:
- x: 1.0
- y: -1.0

Main:
Exercise: Draw a stack diagram for the following program

```java
class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

public static void main(String[] args) {
    Point p = new Point();
    Point p2 = new Point(-4, 3);
    p.add(p2);
}
```
Draw a stack diagram

Function Stack:
Point():
  • this

Created objects
x: 1.0
y: -1.0

Main:
Draw a stack diagram

Function Stack:

Point():
  • this

Created objects:

x: 1.0
y: -1.0

Main:

• p
Draw a stack diagram

Function Stack:

Point():
• this

Main:
• p

Created objects:
• x: 1.0
• y: -1.0
Draw a stack diagram

Function Stack:

Point():
  • this

Created objects

x: 0.0
y: 0.0

Main:

• p
Exercise: Draw a stack diagram for the following program

```java
class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }

    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }

    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

public static void main(String[] args) {
    Point p = new Point();
    Point p2 = new Point(-4, 3);
    p.add(p2);
}
```
Exercise: Draw a stack diagram for the following program

```java
class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }

    public static void main(String[] args) {
        Point p = new Point();
        Point p2 = new Point(-4, 3);
        p.add(p2);
    }
}
```
Draw a stack diagram

Function Stack:

Point():
• this
• inx = -4
• iny = 3

Created objects:

• x: 1.0
• y: -1.0

Main:
• p
Draw a stack diagram

Function Stack:
- Point():
  - this
  - inx = -4.0
  - iny = 3.0

Main:
- p

Created objects:
- x: -4.0
- y: 3.0
- x: 1.0
- y: -1.0
Exercise: Draw a stack diagram for the following program

class Point {
    public double x = 0.0;
    public double y = 0.0;

    public Point() {
        this.x = 1;
        this.y = 1;
    }
    public Point(double inx, double iny) {
        this.x = inx;
        this.y = iny;
    }
    public void sub(Point p) {
        this.x = this.x - p.getX();
        this.y = this.y - p.getY();
    }
}

double p = new Point();
Point p2 = new Point(-4, 3);
p.add(p2);
Draw a stack diagram

**Function Stack:**

**Point():**
- this
- inx = -4.0
- iny = 3.0

**Main:**
- p
- p2

**Created objects**

<table>
<thead>
<tr>
<th>x:</th>
<th>y:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>-4.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Main:
• p
• p2

Function Stack:
Point():
• this
• inx = -4.0
• iny = 3.0

Created objects:

- x: 1.0
  y: -1.0

- x: -4.0
  y: 3.0
Example: Add using a static method

• Make a new static function called “add” that takes in two points, adds their x and y coordinates, and returns a new point
Exercise: Objects and Arrays

Arrays can store objects just like any other type (such as ints, Strings, etc.)

Write a program that asks the user for a number of points and stores them in an array.
Exercise: Draw a stack diagram for the previous program
Access modifiers
Class inheritance
Inheritance: feature for organizing classes into hierarchies
Inheritance: subclasses refine behavior/state
Inheritance

Entity

Player

NPC

Shop Keeper

Quest Giver

Orc

Minion

King
Polymorphism
public class Zoo {
    public static void main(String[] args) {
        Animal animal1 = new Animal();
        animal1.locomote();

        Animal animal2 = new Reptile();
        animal2.locomote();
    }
}

class Animal {
    public Animal() {
    }
    public void locomote() {
        System.out.println("I am moving!");
    }
}

class Reptile extends Animal {
    public Reptile() {
    }
    public void locomote() {
        System.out.println("I am walking!");
    }
}
Exercise: What is the output of this program?

```java
public class Zoo {
    public static void main(String[] args) {
        Animal animal1 = new Animal();
        animal1.locomote();

        Animal animal2 = new Fish();
        animal2.locomote();
    }
}

public class Animal {
    public Animal() {
    }
    public void locomote() {
        System.out.println("I am moving!");
    }
}

public class Fish extends Animal {
    public Fish() {
    }
    public void locomote() {
        System.out.println("I am swimming!");
    }
}
```
Exercise: Implement a Bird animal
OOP Example & Design: Vending machine
OOP Design: Vending machine
Defining the snack class

```java
public class Snack {
    private int mQuantity;
    private double mCost;
    private String mName;

    public Snack(String name, int quantity, double cost) {
        mQuantity = quantity;
        mCost = cost;
        mName = name;
    }

    public String getName() {
        return mName;
    }

    public void buy() {
        if (mQuantity > 0) {
            mQuantity--;
        }
    }
}
```
Testing the Snack class

```java
public static void main(String args[])
{
    Snack snack = new Snack("Slurm", 10, 1.5);
    System.out.println("Snack: " + snack.getName());
}
```
Objects: Stack diagrams revisited

```java
public static void main(String[] args) {
    double userCash = 8.0;
    Snack soda = new Snack("Tang", 10, 1.5); // call constructor
    soda.buy();
}
```
Exercise: draw a stack diagram for this program
Exercise: Define a class BankAccount

BankAccount should have the following data:
• Name
• Amount

BankAccount should have the following operations:

• currentBalance() // returns current amount in the bank account
• withdraw(float amt) // withdraw the given amount from the account
• deposit(float amt) // deposit the given amount to the account