Week 3: Lab 1

CS2030S Lab 16B

Overview

1. Introduction
2. Recap
3. Lab 0 Solution
4. Lab 1 Briefing

1. Introduction

Chrysline Lim, Y2 CS

chrysline.lxh@gmail.com

Alissa, Y3 CS

alissayarmantho1@gmail.com

1. Introduction

Tell me more about yourself!

Name

Year

Hobby/Fun Fact

1. Introduction

Yes

- Guide you in the right direction
- Help you to bridge the learning curve
- Give you good feedback
- Clarify concepts

No

- Be available 24/7
- Debug for you
- Code for you
- Teach from scratch

1. Introduction

Typical lab session

1. Short recap of the week’s content
2. Extra practice questions if any
3. Review of previous lab
4. Briefing about the lab assignment
5. Start on lab (you can leave whenever you want to)

PollEv.com/chryslinelim652

Sample question!

2. Recap: Information Hiding

PollEv.com/chryslinelim652

- All fields must be set to private
- Part of the concept of abstraction
- There are getters and setters to access these fields but you should not use them

2. Recap: Encapsulation

PollEv.com/chryslinelim652

- Group related things together

2. Recap: Tell, Don’t Ask

- Don’t retrieve information from a class; violates abstraction barrier
- Instead ask the class to do something for you with its own information

2. Recap: Inheritance

- Is-A relationship (vs Has-A for Composition)
- First line in constructor has to be super()
- Child <: Parent

```
Circle c = new ColoredCircle(...);
```

PollEv.com/chryslinelim652

2. Recap: Polymorphism

- Inheritance, polymorphism and method overriding are closely related

1. Inheritance: create a parent-child relationship
    - Circle, Square, Triangle inherit from Shape

- Let’s say the Shape class has a method getArea()

2. Override: each child class will override that method with their own implementation
3. Polymorphism: write the code for Shape, and at run-time Circle, Square or Triangle can be substituted in and their corresponding getArea method will be executed instead

PollEv.com/chryslinelim652

3. Lab 0 Solution - Point.java

```
class Point {
  private double x;
  private double y;
  public Point(double x, double y) {
    this.x = x;
    this.y = y;
  }

  public double distanceTo(Point p) {
    return Math.sqrt((this.x - p.x) * (this.x -
p.x) + (this.y - p.y) * (this.y - p.y));
  }

  public String toString() {
    return "(" + this.x + ", " + this.y + ")";
  }
}
```

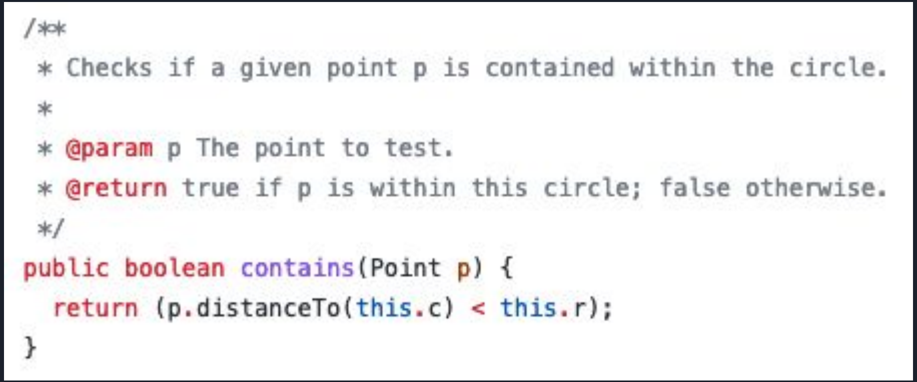
- Keep fields private for **information hiding**
- Use `this` to disambiguate between:
  - Instance fields (this.x)
  - Class field (static)

3. Lab 0 Solution - Point.java

```
class Point {
  private double x;
  private double y;
  public Point(double x, double y) {
    this.x = x;
    this.y = y;
  }

  public double distanceTo(Point p) {
    return Math.sqrt((this.x - p.x) * (this.x -
p.x) + (this.y - p.y) * (this.y - p.y));
  }

  public String toString() {
    return "(" + this.x + ", " + this.y + ")";
  }
}
```

- Expose only the required methods and constructor with `public`
- No getter methods
  - e.g. getX(), getY()
  - Violates tell, don't ask

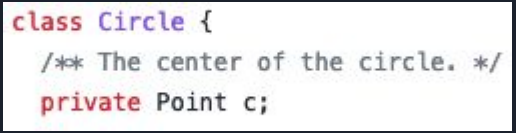
3. Lab 0 Solution - Circle.java



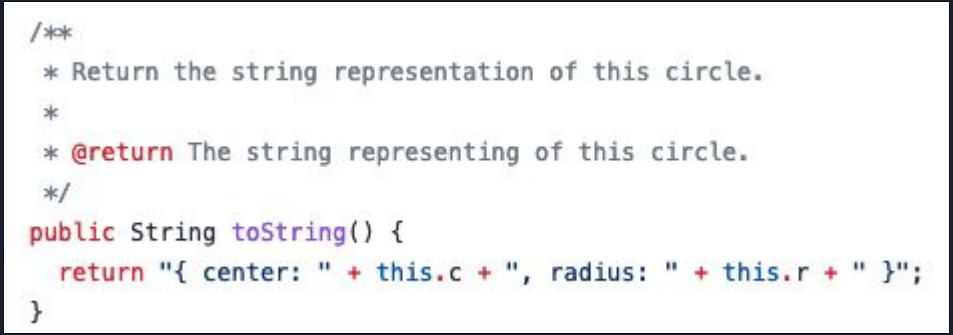
```java
/**
 * Checks if a given point p is contained within the circle.
 *
 * @param p The point to test.
 * @return true if p is within this circle; false otherwise.
 */
public boolean contains(Point p) {
  return (p.distanceTo(this.c) < this.r);
}
```

- Tell, don’t ask

3. Lab 0 Solution - Circle.java



```
class Circle {
  /** The center of the circle. */
  private Point c;
```



```
/**
 * Return the string representation of this circle.
 *
 * @return The string representing of this circle.
 */
public String toString() {
  return "{ center: " + this.c + ", radius: " + this.r + " }";
}
```

- We are telling the centre to print itself, rather than getting the x and y of the centre and printing it.
- Tell, don't ask

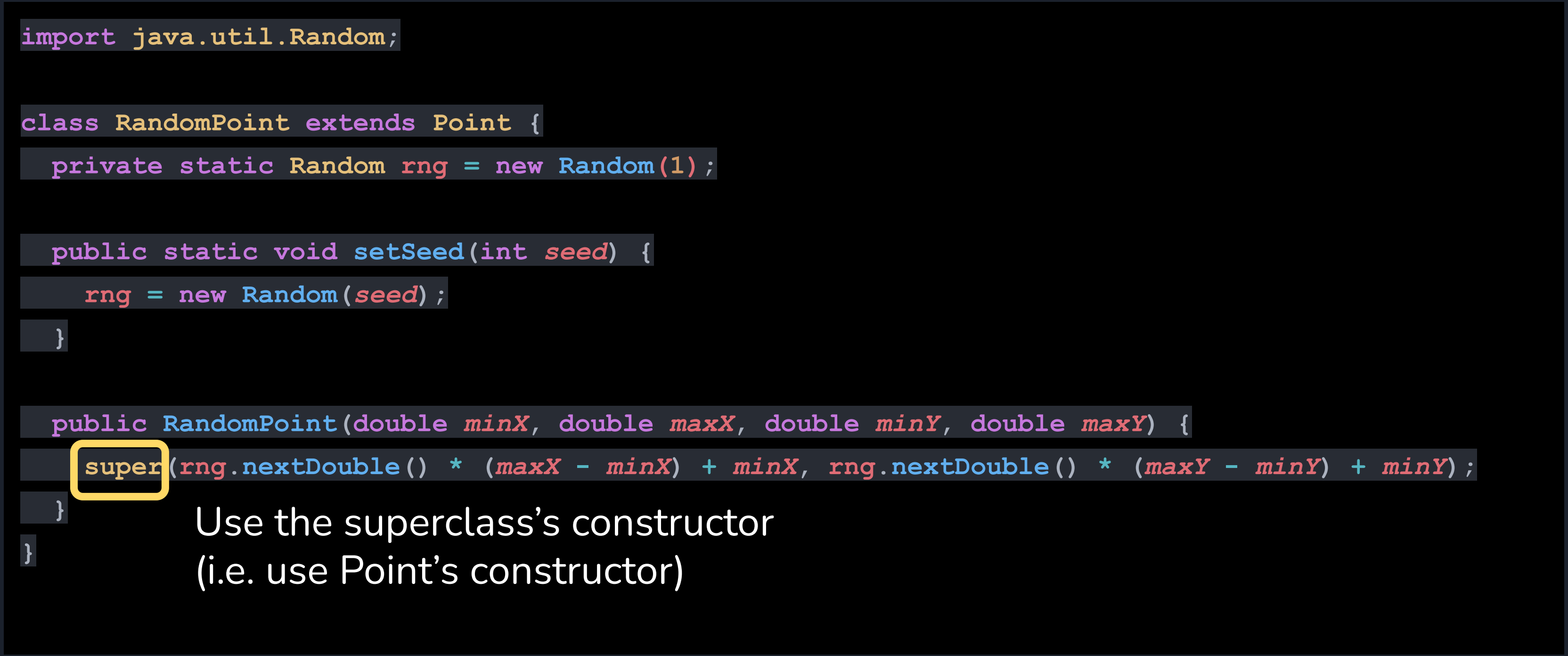
3. Lab 0 Solution - RandomPoint.java

```java
import java.util.Random;

class RandomPoint extends Point {
  private static Random rng = new Random(1);

  public static void setSeed(int seed) {
    rng = new Random(seed);
  }

  public RandomPoint(double minX, double maxX, double minY, double maxY) {
    super(rng.nextDouble() * (maxX - minX) + minX, rng.nextDouble() * (maxY - minY) + minY);
  }
}
```

✅ RandomPoint IS-A Point => Inheritance

❌ Point HAS-A RandomPoint => Composition

3. Lab 0 Solution - RandomPoint.java



```java
import java.util.Random;

class RandomPoint extends Point {
  private static Random rng = new Random(1);

  public static void setSeed(int seed) {
    rng = new Random(seed);
  }

  public RandomPoint(double minX, double maxX, double minY, double maxY) {
    super(rng.nextDouble() * (maxX - minX) + minX, rng.nextDouble() * (maxY - minY) + minY);
  }
}
```

Use the superclass’s constructor
(i.e. use Point’s constructor)

4. Lab 1 Brief: Discrete Event Simulator

- Deadline: 31 August, 2022, 23:59
- Marks: 3%

Goal:

Practice the **basic OOP principles**:

encapsulation, abstraction, inheritance, and polymorphism.

4. Lab 1 Brief: Discrete Event Simulator

Inputs:

n - number of customers

k - number of service counters at the shop

n pairs of double values A1, S1, A2, S2, ... An, Sn

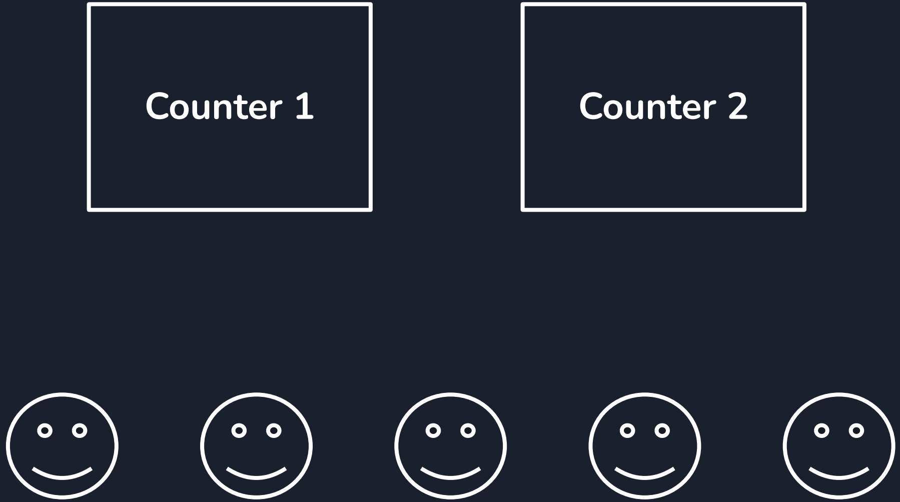
A1 - arrival time of first customer

S1 - service time of first customer

4. Lab 1 Brief: Discrete Event Simulator

inputs:

```
5 2
1.0 1.0
1.2 1.0
1.4 1.0
1.6 1.0
2.1 1.0
```



inputs:

```
5 2
1.0 1.0
1.2 1.0
1.4 1.0
1.6 1.0
2.1 1.0
```

Time	Counter 0	Counter 1	Customer
0.0	available	available	-
1.0	Customer 0 (ends at 2.0)	available	C0 arrives & service begin
1.2		Customer 1 (ends at 2.2)	C1 arrives & service begin
1.4			C2 arrives & departed
1.6			C3 arrives & departed
2.0			C0 service done & departed
2.1	Customer 4 (ends at 3.1)		C4 arrives & service begin
2.2			C2 service done & departed
3.1		available	C4 service done & departed

4. Lab 1 Brief: Discrete Event Simulator

`` `**ShopSimulation.java**` <: `Simulation.java` ``

Process inputs & Initialize ShopEvents

`` `**ShopEvent.java**` <: `Event.java` ``

4 kinds of events: Arrival, Service begin, Service end, Departure

`` `Simulator.java` ``

The driver of the simulation

4. Lab 1: Simulation I: Encapsulation

Objectives:

Encapsulation to group relevant fields and methods into new classes

Inheritance and composition to model the relationship between the classes

Information hiding to hide internal details (correctly use access modifiers)

Using **polymorphism** to make the code more succinct and extendable in the future, while adhering to LSP

4. Lab 1: Simulation I: Encapsulation

What are the nouns? These are good candidates for new classes.

e.g. Customer? Shop? And more ...

For each class, what are the attributes/properties relevant to the class? These are good candidates for fields in the class.

e.g. Customer has id, arrival time, service time, etc.

Do the classes relate to each other via IS-A or HAS-A relationship?

e.g. ShopEvent and Event. What else?

4. Lab 1: Simulation I: Encapsulation

For each class, what are their responsibilities? What can they do?
These are good candidates for methods in the class.

e.g. A shop need to manage the availability of counters, etc.

How do the objects of each class interact?
These are good candidates for public methods.

e.g. When a customer arrives, Shop provides information on if there is any available counters, etc.

What are some behavior that changes depending on the specific type of objects?

e.g. If a customer will be served depends on the availability of counters

That’s all for today! Thanks for coming!

Join the Telegram group! https://t.me/+Hr85eZ8CGMg3MzU1

Feedback form:

https://docs.google.com/forms/d/e/1FAIpQLSck-bEpWfvRiEp4Nh_Pb2loof9NOTzSRfj3OJA-42dN_hAc2g/viewform?usp=sf_link

QR Codes

Telegram

Slides

Feedback

Slides adapted from Chris

Additional Things: Vim Plugins + Skins (OPTIONAL)

https://github.com/alissayarmantho/myVimConfig

1. Go to the stu comp nodes via ssh
2. Run:
   a. git clone https://github.com/alissayarmantho/myVimConfig.git
   b. cd myVimConfig
   c. cp .vimrc ~
   d. cp -r .vim ~

Important Vim Commands:

i => Start typing

ESC => Type commands (:wq to save and quit, :q to quit without saving, :w to just save)

Additional things: Setting up ssh without password prompts (OPTIONAL)

https://kb.iu.edu/d/aews

Follow the step 1-9 of `Set up public key authentication using SSH on a Linux or macOS computer` instructions

On step 2, use default file name (just enter) and don’t put in any password when prompted (just enter)

Git Bash is a good alternative to windows command prompt if it gi

Additional things: How to scp your files to sunfire (OPTIONAL)

In your folder that contains your Lab1 folder (from the unzip)

```
scp -r ./Lab1 <name>@sunfire.comp.nus.edu.sg:<directory in sunfire>
```

eg:
```
scp -r ./Lab1 alissa@sunfire.comp.nus.edu.sg:~
```