CS2030S Recitation

Week 3: Problem Set 1

brian

2026-01-27

National University of Singapore

Introduction

Attendance

About me

- Brian Cheong
- Studied CS in NUS for UG
- Was a UG Lab TA for CS2030S since year 2
- Now 2nd year PhD candidate

Contact me

- Website: https://www.comp.nus.edu.sg/~bskch
- Email: bskch@nus.edu.sg

Flow of Recitation

Flow of Recitation

- Please watch lectures before coming
- Don't have to complete recitation sheet but at least read the questions
- Discussion based
- Office hours TBD but feel free to email me to arrange

Recap

Stack and Heap diagram

Stack

- Made of frames
  - Contains bindings between variable names and its value
- These frames are for *active* method invocation frames
- 1 method invocation $\rightarrow$ 1 frame
- When the method is done, remove the frame

Heap

- Where objects that are created live
- Objects contain information about that instance
  - mainly fields (for now...)
- Why do we need the heap?
  - To allow objects to "live" on after stack frame that created it is destroyed

Problem Set

Q1a

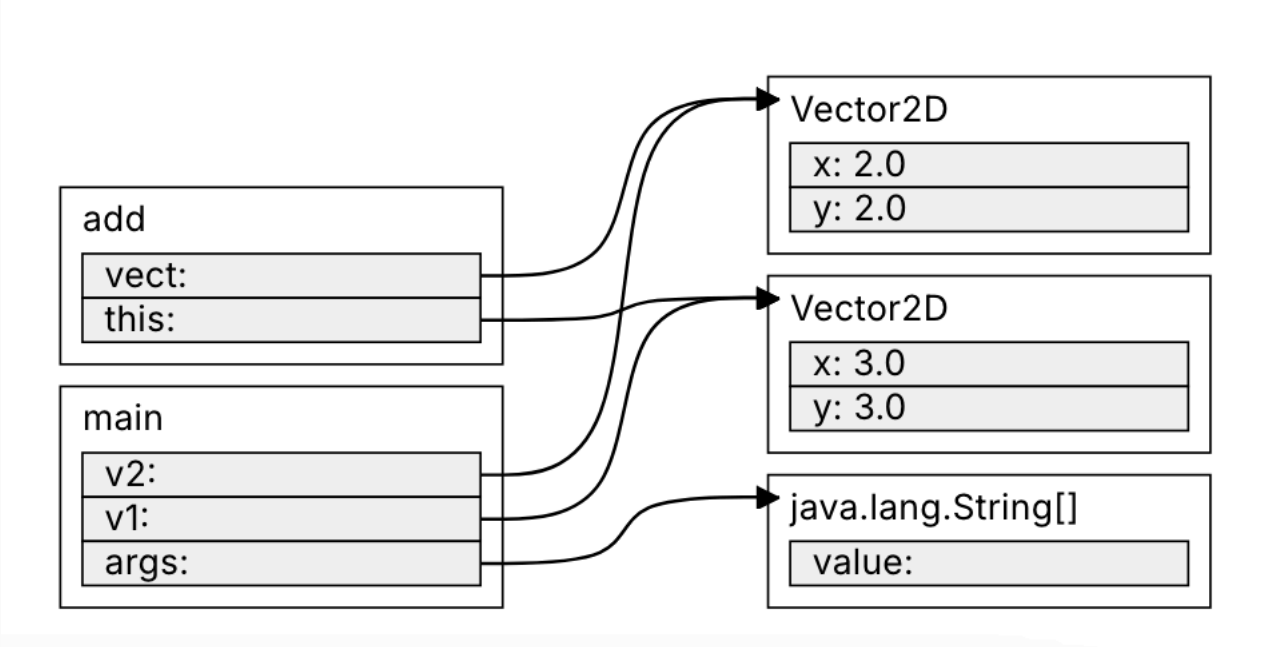
```
1  class Vector2D {
2    private double x;
3    private double y;
4    public Vector2D(double x, double y)
     {
5      this.x = x;
6      this.y = y;
7    }
8    public void add(Vector2D vect) {
9      this.x = this.x + vect.x;
10     this.y = this.y + vect.y;
11     // Line A
12   }
13 }
```

Suppose we run

```
1  Vector2D v1 = new Vector2D(1,
   1);
2  Vector2D v2 = new Vector2D(2,
   2);
3  v1.add(v2);
```

- What would the stack and heap look like at `Line A`

Q1a


add
vect:
this:
main
v2:
v1:
args:
Vector2D
x: 2.0
y: 2.0
Vector2D
x: 3.0
y: 3.0
java.lang.String[]
value:

Q1b

Suppose that `x` and `y` is now

```
1  class Vector2D {
2    private double[] coord2D;
3    // : other code omitted
4  }
```

1. What else would have to change in `Vector2D`?
2. Would the program fragment in `main` still be valid?

1. This is a reminder for brian to showcase live
2. Change wherever `this.x` and `this.y` is used
   - `this.x` → `this.coord2D[0]`
   - `this.y` → `this.coord2D[1]`
3. Yes it would still be valid
   - The changes to `Vector2D`'s internals are hidden behind the abstraction barrier

Recap: Dynamic Binding

Recap: Dynamic Binding

- Compile Time Type
  - The type you give to a variable
  - E.g. `Animal animal = new Dog()` CTT(`animal`) = `Animal`
- Run Time Type
  - The type of the object that actually lives in the heap
  - E.g. `Animal animal = new Dog()` RTT(`animal`) = `Dog`

Recap: Dynamic Binding

- Polymorphism (greek for many forms)
  - Do the “same” action differently based on run time type
  - All humans can walk, but they all walk differently
- Dynamic binding is the mechanism to achieve polymorphism
  - let the runtime type decide which method implementation is invoked
  - Done in a two step process
    1. compile time step
    2. run time step

2 Step Dynamic Binding Overview

- During compilation
  - To figure out which method descriptor (method signature + return type) to use at run time
  - Happens during compilation so can only use compile time type
- During run time
  - Based on the run time type, find the method that matches the descriptor

Compile time step

Use `a.foo(b)` and `param` as the parameter to foo as a running example

1. See what methods CTT(a) has
   - may have multiple overloaded `foo` methods
2. See which of these methods have CTT(b) <: CTT(param)
3. If there are still multiple options, choose the most specific method
   - arguments to a method M are more specific if they can be passed to a method N without compile error
   - Intuitively take the "smaller" one (subtyping is a good approximation)
4. Method descriptor stored in bytecode for run time

Run time step

Use same example `a.foo(b)` and `param`

1. Set RTT(a) as the current class you look at and the descriptor M
2. Look for *exactly* M in the current class
3. Found it? Great you execute that method M
4. No find? Go to the super of the current class and jump back to step 2.

Q2: background

Point.java

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

Q2: background

Circle.java

```
1  public class Circle {
2    private Point centre;
3    private int radius;
4    public Circle(Point centre, int radius) {
5      this.centre = centre;
6      this.radius = radius;
7    }
```

Q2: background

continued...

```
8   public boolean equals(Object obj) {
9     System.out.println("equals(Object) called");
10    if (obj == this) {
11      return true;
12    }
13    if (obj instanceof Circle) {
14      Circle circle = (Circle) obj;
15      return (circle.centre.equals(centre) && circle.radius == radius);
16    } else {
17      return false;
18    }
19  }
```

Q2: background

continued...

```
20    public boolean equals(Circle circle) {
21      System.out.println("equals(Circle) called");
22      return circle.centre.equals(centre) && circle.radius == radius;
23    }
24  }
```

Q2: background

We have the following code fragment

```
1  Circle c1 = new Circle(new Point(0, 0), 10);
2  Circle c2 = new Circle(new Point(0, 0), 10);
3  Object o1 = c1;
4  Object o2 = c2;
```

Q2a(i)

In `Circle.java`

```
1  equals(Object)
2  equals(Circle)
```

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

`o1.equals(o2);`

Target CTT	Target Methods	Arg CTT	Method	Target RTT	Implementation
`Object`	`equals(Object)`	`Object`	`equals(Object)`	`Circle`	`Circle's equals(Object)`

Q2a(ii)

In `Circle.java`

```
1  equals(Object)
2  equals(Circle)
```

`o1.equals((Circle) o2);`

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

Target CTT	Target Methods	Arg CTT	Method	Target RTT	Implementation
`Object`	`equals(Object)`	`Circle`	`equals(Object)`	`Circle`	`Circle's equals(Object)`

Q2a(iii)

In `Circle.java`

```
1  equals(Object)
2  equals(Circle)
```

`o1.equals(c2);`

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

Target CTT	Target Methods	Arg CTT	Method	Target RTT	Implementation
`Object`	`equals(Object)`	`Circle`	`equals(Object)`	`Circle`	`Circle's equals(Object)`

Q2a(iv)

In `Circle.java`

```
1  equals(Object)
2  equals(Circle)
```

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

`c1.equals(o2);`

Target CTT	Target Methods	Arg CTT	Method	Target RTT	Implementation
`Circle`	`equals(Object)` `equals(Circle)`	`Object`	`equals(Object)`	`Circle`	`Circle's` `equals(Object)`

Q2a(v)

In `Circle.java`

```
1  equals(Object)
2  equals(Circle)
```

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

`c1.equals((Circle) o2);`

Target CTT	Target Methods	Arg CTT	Method	Target RTT	Implementation
`Circle`	`equals(Object)` `equals(Circle)`	`Circle`	`equals(Circle)`	`Circle`	`Circle's` `equals(Circle)`

Q2a(vi)

In `Circle.java`

```
1  equals(Object)
2  equals(Circle)
```

`c1.equals(c2);`

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

Target CTT	Target Methods	Arg CTT	Method	Target RTT	Implementation
`Circle`	`equals(Object)` `equals(Circle)`	`Circle`	`equals(Circle)`	`Circle`	`Circle's` `equals(Circle)`

Q2b

In essence...

```
1  equals(Object obj) {
2    :
3  }
4  equals(Circle circle) {
5    :
6  }
```

```
1  Circle c1 = new Circle(...);
2  Circle c2 = new Circle(...);
3  Object o1 = c1;
4  Object o2 = c2;
```

1. What is the return value of `c1.equals(c2)`? Explain
   - Which method gets invoked?
   - What's the return value?
   - `equals(Circle)`
   - `false` since `Point` did not implement `equals`

The End

See you next week