

Polymorphism

Preliminary

The following questions are not going to be discussed. Answers is given without explanation. You are encouraged to discuss them on Ed.

1. Consider the following definition of `Vector2D` class.

Vector2D

```
class Vector2D {
  private double x;
  private double y;

  public Vector2D(double x, double y) { // (1)
    this.x = x;
    this.y = y;
  }

  public void add(Vector2D vect) {      // (2)
    // Line A
    this.x = this.x + vect.x;
    this.y = this.y + vect.y;
    // Line B
  }
}
```

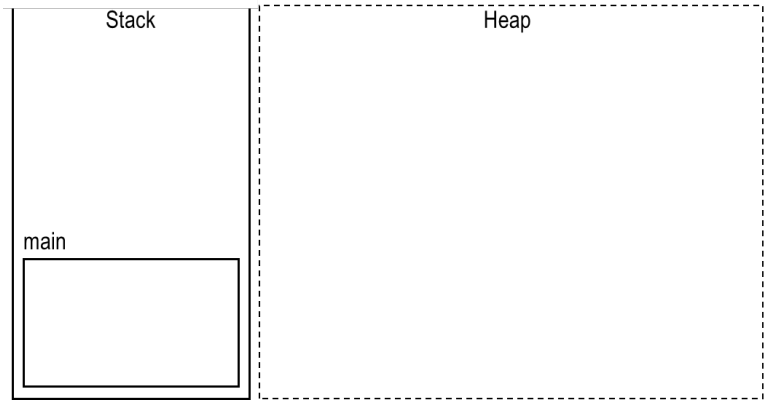
Suppose that the following program fragment is in the `main` method.

Code Snippet

```
Vector2D v1 = new Vector2D(1, 1); // (3)
Vector2D v2 = new Vector2D(2, 2); // (4)
v1.add(v2);                       // (5)
// Line C
```

(a) Provide the sequence of execution using the label (1) to (5) until the execution reaches the line labeled `// Line C`.

(b) Show the content of the stack and the heap when the execution reaches the line labeled `// Line A` above. You may use the following template.



(c) Show the content of the stack and the heap when the execution reaches the line labeled `// Line B` above.

(d) Show the content of the stack and the heap when the execution reaches the line labeled `// Line C` above.

(e) Suppose that the representation of `x` and `y` have been changed to a **`double`** array.

Vector2D with Array

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

- What changes do you need for the other parts of the class `Vector2D`?
- Would the code snippet above still be valid?

2. Consider the following classes.

Class Hierarchy

```
class A {
  public void f(B param) { }
  public void f(C param) { }
}
class B extends A {
  public void f(A param) { }
  public void f(B param) { }
}
class C extends B {
  public void f(C param) { }
  public void f(B param) { }
}
```

Further consider the following code snippet.

Code Snippet

```
A a = new C();
B b = new B();
```

For each of the following method invocation, answer the following questions.

(i) What is compile-time type of the target of invocation?
(ii) What is compile-time type of the argument?
(iii) What are the accessible methods?

(iv) What are the applicable methods?
(v) What is the most specific method?
(vi) What is the method invoked?

(a) `b.f(a);`
(b) `a.f(b);`

Discussion

The following questions are problem sets to be discussed during recitation. You are expected to attempt them before coming to the recitation. You may be randomly called to present your answer during recitation. You are encouraged to discuss them on Ed.

1. Consider the following classes.

Bag

```
class Bag {
  private int item;

  Bag(int item) {
    this.item = item;
  }

  public int retrieve() {
    return this.item;
  }

  public void store(int item) {
    this.item = item;
    // Line A
  }
}
```

Program

```
class Program {
  static Bag merge(Bag b1, Bag b2) {
    int sum = b1.retrieve() + b2.retrieve();
    b1.store(sum);
    return b1;
  }

  public static void main(String[] args) {
    Bag b1 = new Bag(1);
    Bag b2 = new Bag(3);
    b1 = merge(b1, b2);
  }
}
```

Show the content of the stack and the heap when the execution reaches the line labeled `// Line A` above. Ignore `args` in your diagram.

2. Study the following `Point` and `Circle` classes.

Point

```
public class Point {
  private double x;
  private double y;

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

Circle

```
public class Circle {
  private Point center;
  private int radius;

  public Circle(Point center, int radius) {
    this.center = center;
    this.radius = radius;
  }

  @Override
  public boolean equals(Object obj) {
    System.out.println("equals(Object) called");
    if (obj == this) {
      return true;
    }
    if (obj instanceof Circle) {
      Circle circle = (Circle) obj;
      return (circle.center.equals(center) && circle.radius == radius);
    } else {
      return false;
    }
  }

  public boolean equals(Circle circle) {
    System.out.println("equals(Circle) called");
    return circle.center.equals(center) && circle.radius == radius;
  }
}
```

You are also given the following variable declarations.

Code Snippet

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

(a) What is the return value of `c1.equals(c2)`? Explain.

(b) For each of the statement below, trace through the two-step dynamic dispatch process to show which `equals` method is invoked during run-time.

(i) `o1.equals(o2);`
(ii) `o1.equals((Circle) o2);`
(iii) `o1.equals(c2);`
(iv) `c1.equals(o2);`
(v) `c1.equals((Circle) o2);`
(vi) `c1.equals(c2);`

Hint: You can read the Java Language Specification regarding method invocation to understand more. We have already covered them in lecture and notes. However, the interaction with other components may require revisiting the specification.

Exercise

The following questions are to be discussed on Ed as no answer will be given. ***Use only concepts that you have learnt****. You are expected to attempt them before the next lecture as the concepts may be used and/or mentioned in the lecture. The concept may also be tested for assessment.*

1. Consider the `Vector2D`, `Point`, and `Circle` classes in the recitation.

   We have an array of items that may be an instance of one of the classes above. Recap that one of the basic idea in achieving code maintainability is abstraction. In this case, we want to have a single implementation that can be used for any of the classes above.

   What we want to do is to implement a function that returns `true` if the instance `obj` is inside an array called `array`. Here, inside means that there is an instance that is equal to `obj`. This function should work for `Vector2D`, `Point`, and `Circle` classes. Since the only common superclass of the three classes above is `Object`, our implementation is as follows. Luckily, `Object` has the method `equals`. Note that we use the enhanced for-loop[1] in our implementation.

   Contains Method

```
class Util {
  public static boolean contains(Object[] array, Object obj) {
    for (Object curr : array) {
      if (curr.equals(obj)) {
        return true;
      }
    }
    return false;
  }
}
```

   (a) Show that the code works for `Vector2D`, `Point`, and `Circle` classes by writing test cases.
   (b) Does the code work for other classes (e.g., `String`)? If yes, write the test cases for it. Otherwise, describe how you would support other classes.
   (c) Does the code work for primitive types (e.g., `int`)? If yes, write the test cases for it. Otherwise, describe how you would support primitive types in `contains`.

   Hint: You may check how Java `Arrays` [2] supports primitive.
   (d) Is there a simpler way to support both primitive type and reference type using the same code?

Bibliography

[1] Oracle, “For-Loop.” [Online]. Available: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html

[2] Oracle, “Java 25 API: Arrays.” [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/util/Arrays.html

[3] Oracle, “Array.” [Online]. Available: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

[4] Oracle, “Compile-Time Step 2: Determine Method Signature.” [Online]. Available: https://docs.oracle.com/javase/specs/jls/se25/html/jls-15.html#jls-15.12.2

[5] Oracle, “Method Signature.” [Online]. Available: https://docs.oracle.com/javase/specs/jls/se25/html/jls-8.html#jls-8.4.2