CSE 11
Accelerated Intro to Programming
Discussion Section 4

Sachin Deshpande
Summer 1 2021

This discussion is being recorded

Logistics

- PA3 is due today 11:59 pm
- PA4 released today
- Resubmissions! 75%

Interface

- Classes that share behavior
- An interface is declared by using the interface keyword
- all the methods in an interface are declared with the empty body, For Example-

```
interface /* Interface Name */ {
    /* Method Signature */;
    /* Method Signature */;
    /* ... */
}
```

- classes and abstract classes can implement interfaces with the following syntax -

```
class /* Class Name */ implements /* Interface Name */ {
    /* ... */
}
```

- interface allowed us to treat multiple classes as a shared type, For Example - We use it to create Unions of regions without worrying about what the underlying Region type actually was
- All methods in interface are implemented

Example:

```
interface Region {
    boolean contains(Point p);
}
class SquareRegion implements Region {
    ...
    public boolean contains(Point toCheck) {...}
}
class CircleRegion implements Region {
    ...
    public boolean contains(Point toCheck) {... }
}
class UnionRegion {
    Region r1, r2;
    UnionRegion(Region r1, Region r2) {
        this.r1 = r1;
        this.r2 = r2;
    }
    public boolean contains(Point p) {
        return this.r1.contains(p) || this.r2.contains(p);
    }
}
```

A class that implements an interface can have its own fields and methods

different fields and different methods

Point center;
int length;

Point center;
int radius;

implements Region

r1 and r2 can belong to any class that implements Region

```
Class Example {
    SquareRegion r1 = new ...();
    CircleRegion r2 = new ...();
    UnionRegion u = new UnionRegion(r1, r2);
    Point p = ...();
    u.contains(p);
}
```

Tester → a library that allows you to test your code

- import tester.*;
  - tester.jar – java archive
    - Libraries that contain classes that we can use in our own code
      - Tester
- Tester class allows us to create methods to unit test our code
  - Unit testing – compare actual values versus expected values
    - t.checkExpect(<actual value>, <expected value>); → call the method, → reference
    - test method name should begin with "test"
  - Goal: get all tests to pass
    - Confidence that your code/solution is correct

```
boolean test---( Tester t) {
  return t.checkExpect(actual, expected);
}
```

Tester

```
import tester.*;
class Test1 {
	int method1() {}

	//week1 - test as a single method to a field
	int field1 = this.method1();

	//wed of Week2
	boolean testMethod1(Tester t) {
		return t.checkExpects(this.method1(), 1);
	}

	 //week 3 after we discuss main()
	 void testMethod2(Tester t) {
		t.checkExpects(this.method1(), 1);
	 }
}
```

PA4

Different assignments in this course have different collaboration policies. On this assignment, you cannot share code publicly on Piazza or with any other students in the course. If you need to share code, ask in a private Piazza post or in 1-on-1 hours. Still do ask any public code questions about lecture code, quizzes that are past, or other conceptual questions, just no code from this PA.

Thanks!