CSE 11
Accelerated Intro to Programming
Discussion Section 10

Sachin Deshpande
Summer 1 2021

This discussion is being recorded

Logistics

no resubmit

- PA9 due today at 11:59PM
- FINAL EXAM!!!
    - See Piazza for details
    - Like a midterm, 5%/10%/15%

Overriding methods

- Providing a different implementation of an existing method
- The method's header must be identical to the method in the superclass. The body can be different

```
// Base Class
class Parent {
    void show()
    {
        System.out.println("Parent's class");
    }
}
  // Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    @Override
    void show()
    {
        System.out.println("Child's class");
    }
}
```

Overloading :

- Same class
- Two or more methods
  - ↳ Share the same method name
  - ↳ Different parameters (number and/or type different)

Overriding

- Different classes (one extends the other)
- Two or more methods
- Exact same signature (Return type, method name, parameters are same)

The instanceof operator

- It is used between an object and the name of a class, and returns true if that object's type is equal to or is a subclass of that class.

boolean var = objName instanceof ClassName;

object

Class

```
class A {}
class B extends A {}
class C extends A {}

A a = new A();
B b = new B();
C c = new C();
A a2 = new B();
boolean isAAnInstanceOfA = a instanceof A; // true
boolean isBAnInstanceOfA = b instanceof A; // true
boolean isBAnInstanceOfB = b instanceof B; // true
```

a instanceof B; // false

a2 instanceof B; // true

Casting

- To treat an instance as having the type of another class
- Only works if `instanceof` evaluates to true

```
class A{}
class B extends A {
    int x;
}

A a = new A();
A a2 = new B();
int x = a2.x; // ???
B b2 = (B) a2;
int x2 = b2.x; // ???
```

a2 instanceOf B; // true
a instanceOf B; // false

Class2 objName = (Class2) obj2;

Summary: Two basic rules to memorize

1. At compile time, the compiler uses the type of the variable to determine what methods can be called
2. At runtime, Java uses the type of the object to determine what method actually runs

Access Modifiers

The public, protected and private access modifiers to clearly indicate the access to different classes, fields and methods.

- The public modifier allows access anywhere.
- The protected modifier allows access anywhere within the same package, or in any of the subclasses of the protected class.
  folder
- The private modifier only allows access within the class that contains them.
- No modifier allows access anywhere within the same package. (You might hear this referred to as "package visibility" or being "package private")

Final Exam

- Similar to Exam1 and 2 (programming + video)
- Cumulative
- Read instructions carefully
- Follow instructions closely

```
int[] taskTwoTest = mystery(/* argument */);
```

Thanks!