

CSE 11


Accelerated Intro to Programming


Discussion Section 10

Shihua Lu, Spring 2021

This discussion is being recorded

Logistics

- PA9  no resubmit due tomorrow at 11:59PM
- Exam
 - Release: 8am Wednesday June 9, 2021
 - Due: 11:59pm Friday June 11, 2021

PA 6 / 7 / 8  90%

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.

```
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
```

boolean var = objName instanceof ClassName;

objName *ClassName*

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) obj1;

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 C) taskTwoTest = mystery (/* argument */);
```

Thanks!