Last Time

• Design of three classes: Human, Computer, and Game
• Implementation of the Human class
• Implementation of the Computer and Game classes
• Implementation of the Player class, which we set as a superclass of the Human and CPU classes
Polymorphism

• A reference variable can take on different forms, either of the declared class or of any related subclass:

```
Player human, cpu;
human = new Human();
cpu = new CPU();
```

• Unfortunately, the `human` variable can’t access any methods exclusive to the `Human` class.
  – Solution: Abstract classes
Abstract

- An abstract class contains some methods that are declared but not implemented.
  - Methods that are not implemented are abstract methods.
  - Methods that are implemented are concrete methods.

- By making the Player class abstract and adding the missing method declarations, we can now make our Human and CPU variables polymorphic.
Dynamic Dispatch

• So now we have a variable `computerPlayer`, which is declared as type `Player` and initialized as type `Computer`. Which `increaseScore()` function gets called?
  – Java decides at runtime to call the version of the method that is most specific to the initialized type rather than the declared type.
  – (Our CPU is still cheating with double points.)
instanceof

• Need to check whether the Player object you created is a Human or a CPU?

```java
if (var1 instanceof Human) {
    // do stuff
} else if (var1 instanceof Computer) {
    // do other stuff
} //if-else
```
Any Questions?