Northeastern University

Lecture 3: Java Review - 3

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Materials are edited by Prof. Jones Yu from

Data Structures and Abstractions with Java, 5th edition. By Frank M. Carrano and Timothy M. Henry.
ISBN-13 978-0-13-483169-5

Outline

- Java Interfaces

Java Interfaces

Java Interfaces

- Program component that declares a number of public methods
  - » Should include comments to inform programmer
  - » Any data fields here should be **`public`**, **`static`**, **`final`**

Listing 2-1 Interface `Measurable`

```
/** An interface for methods that return
the perimeter and area of an object.
*/
public interface Measurable
{
   /** Gets the perimeter.
   @return The perimeter. */
   public double getPerimeter ();
   /** Gets the area.
   @return The area. */
   public double getArea ();
} // end Measurable
```

Listing 2-2 Interface `NamedMeasurable`

```
/** An interface for a class of names. */
public interface NameInterface
{
   /** Sets the first and last names.
   @param firstName A string that is the desired first name.
   @param lastName A string that is the desired last name. */
   public void setName (String firstName, String lastName);

   /** Gets the full name.
   @return A string containing the first and last names. */
   public String getName ();
   public void setFirst (String firstName);
   public String getFirst ();
   public void setLast (String lastName);
   public String getLast ();
   public void giveLastNameTo (NameInterface aName);
   public String toString ();
} // end NameInterface
```

Implementing an Interface

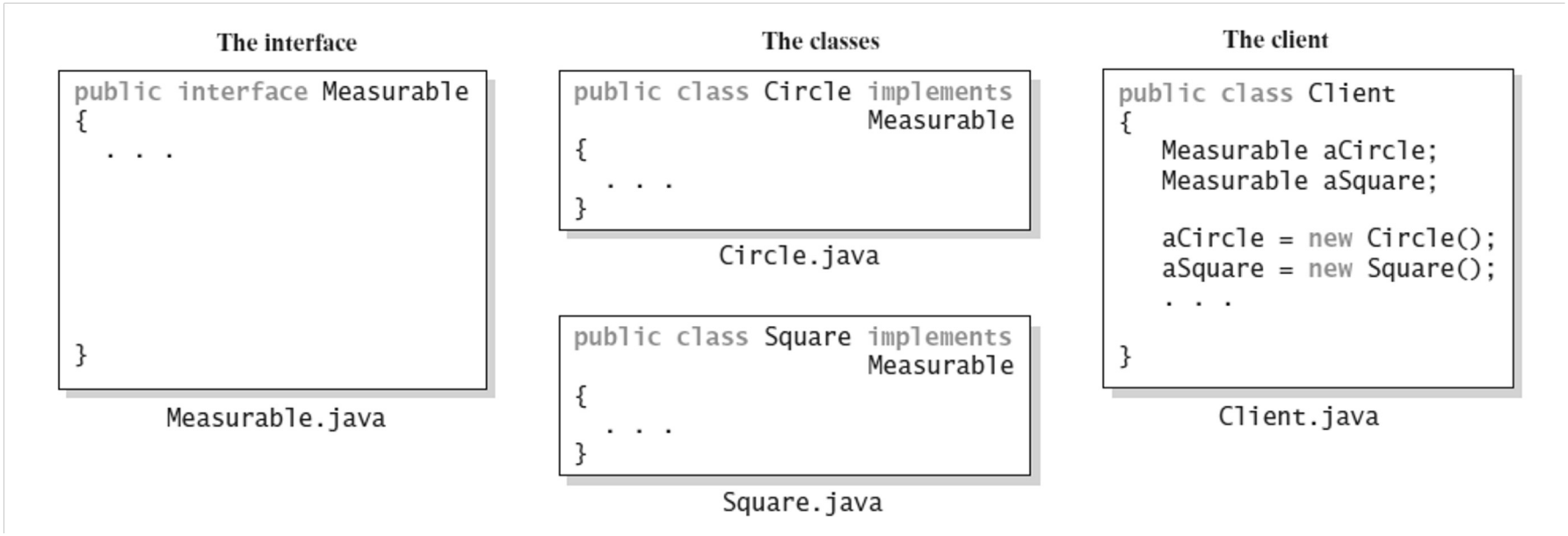


Figure P-3: The files for an interface, a class that implements the interface, and the client

Implementing an Interface

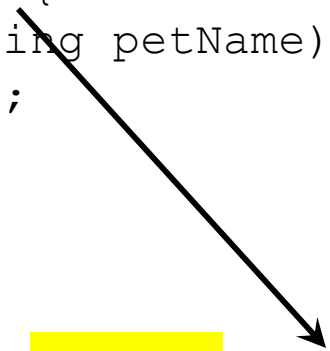
- A way for programmer to guarantee a class has certain methods
- Several classes can implement the same interface
- A class can implement more than one interface

Interface as a Data Type

- You can use a Java interface as a data type
- Indicates variable can invoke certain set of methods and only those methods.
- An interface type is a reference type
- An interface can be used to derive another interface by using inheritance

Interface as a Data Type

```
public interface Nameable {
  public void setName(String petName);
  public String getName();
}
```



```
public interface Callable extends Nameable {
  public void come(String petName);
}
```

Interface vs. Abstract Class

- Purpose of interface similar to that of abstract class
  - » But an interface is *not* a class (or say interface is a special class)
- Use an abstract class …
  - » If you want to provide a method definition
  - » Or declare a private data field that your classes will have in common
- A class can implement several interfaces but can extend only one abstract class.

Interface vs. Abstract Class

- If a class is abstract, you cannot create objects of that class; it can be used only as a base class for other classes.

Exercise

- Write a Java interface (`StudentInterface.java`) that specifies and declares methods for a class of `students`.

Answer

```
public interface StudentInterface {
  public void setStudent(Name studentName, String studentId);
  public void setName(Name studentName);
  public Name getName();
  public void setId(String studentId);
  public String getId();
  public String toString();
}
```

StudentInterface.java

Answer

```
public class Name {
  private String firstName;
  private String lastName;

  public Name(String first, String last){
    firstName = first;
    lastName = last;
  }

  public String toString(){
    return lastName + "," + firstName;
  }
}
```

Name.java

Exercise

- Begin the definition of a class (`Student.java`) that implements the interface that you wrote in answer to the previous question.
- Include data fields, a constructor, and at least one method definition.
- Also, please write a driver program (`StudentTest.java`) to verify your implementation.

Answer

```
public class Student implements StudentInterface{
  private Name fullName;
  private String id;

  public Student(){
    fullName = new Name("","");
    id = "";
  }

  public Student(Name studentName,
                 String studentId){
    fullName = studentName;
    id = studentId;
  }

  public void setStudent(Name studentName,
                         String studentId){
    setName(studentName);
    setId(studentId);
  }

  public void setName(Name studentName){
    fullName = studentName;
  }

  public Name getName(){
    return fullName;
  }

  public void setId(String studentId){
    id = studentId;
  }

  public String getId(){
    return id;
  }

  public String toString(){
    return id + " " + fullName.toString();
  }

} // End of Student class
```

Student.java

Answer

```
public class StudentTest {
  public static void main(String[] args){

    Name student1Name = new Name("Richard", "Anderson");
    String student1Id = "W123456789";

    Student student1 = new Student(student1Name, student1Id);

    System.out.println(student1.getId() + " " + student1.getName());

    // ...
    // You can create another student and use setName() and setId()
  }
}
```

StudentTest.java