CSE 12 – Basic Data Structures and Object-Oriented Design
Lecture 3

Greg Miranda, Spring 2021

This lecture is being recorded

Announcements

- Quiz 3 due Monday @ 12pm
- Survey 1 due tonight @ 11:59pm
- Discussion starts today @ 1pm

Topics

- Questions on Lecture 3?
- Interfaces

13) Given the following definitions:

```
public interface Printable
{
  public abstract String print( boolean duplex );
}
```

```
class Thing1 implements Printable
{
  private String str;

  public Thing1()
  {
    this.str = "Thing 1";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print()
  {
    // print single sided by default
    return this.print( false );
  }
}
```

```
class Thing2 implements Printable
{
  private String str;

  public Thing2()
  {
    this.str = "Thing 2";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print( String user )
  {
    System.out.print( user + ": " );

    // print double sided by default
    return this.print( true );
  }
}
```

And the following variable definitions:

```
Thing1 thing1 = new Thing1();
Thing2 thing2 = new Thing2();
Printable printable;
```

<u>Hint</u>: What does the compiler know about any reference variable at compile time (vs. run time)?

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error" and assume that line is commented out when run.

```
System.out.println( thing1.print() );                 ______________________

System.out.println( thing1.print( "CS11SZZ" ) );      ______________________

System.out.println( thing1.print( false ) );          ______________________
```

13) Given the following definitions:

```
public interface Printable
{
  public abstract String print( boolean duplex );
}
```

```
class Thing1 implements Printable
{
  private String str;

  public Thing1()
  {
    this.str = "Thing 1";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print()
  {
    // print single sided by default
    return this.print( false );
  }
}
```

```
class Thing2 implements Printable
{
  private String str;

  public Thing2()
  {
    this.str = "Thing 2";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print( String user )
  {
    System.out.print( user + ": " );

    // print double sided by default
    return this.print( true );
  }
}
```

And the following variable definitions:

```
Thing1 thing1 = new Thing1();
Thing2 thing2 = new Thing2();
Printable printable;
```

<u>Hint</u>: What does the compiler know about any reference variable at compile time (vs. run time)?

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error" and assume that line is commented out when run.

```
System.out.println( thing2.print() );                 ______________________

System.out.println( thing2.print( "CS11SZZ" ) );      ______________________

System.out.println( thing2.print( false ) );          ______________________
```

13) Given the following definitions:

```
public interface Printable
{
  public abstract String print( boolean duplex );
}
```

```
class Thing1 implements Printable
{
  private String str;

  public Thing1()
  {
    this.str = "Thing 1";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print()
  {
    // print single sided by default
    return this.print( false );
  }
}
```

```
class Thing2 implements Printable
{
  private String str;

  public Thing2()
  {
    this.str = "Thing 2";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print( String user )
  {
    System.out.print( user + ": " );

    // print double sided by default
    return this.print( true );
  }
}
```

And the following variable definitions:

```
Thing1 thing1 = new Thing1();
Thing2 thing2 = new Thing2();
Printable printable;
```

<u>Hint</u>: What does the compiler know about any reference variable at compile time (vs. run time)?

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error" and assume that line is commented out when run.

```
printable = thing1;

System.out.println( printable.print( true) );              ____________________

System.out.println( printable.print() );                   ____________________

System.out.println( printable.print( "CS11SZZ" ) );        ____________________
```

13) Given the following definitions:

```
public interface Printable
{
  public abstract String print( boolean duplex );
}
```

```
class Thing1 implements Printable
{
  private String str;

  public Thing1()
  {
    this.str = "Thing 1";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print()
  {
    // print single sided by default
    return this.print( false );
  }
}
```

```
class Thing2 implements Printable
{
  private String str;

  public Thing2()
  {
    this.str = "Thing 2";
  }

  public String print( boolean duplex )
  {
    return this.str + " duplex = " + duplex;
  }

  public String print( String user )
  {
    System.out.print( user + ": " );

    // print double sided by default
    return this.print( true );
  }
}
```

And the following variable definitions:

```
Thing1 thing1 = new Thing1();
Thing2 thing2 = new Thing2();
Printable printable;
```

<u>Hint</u>: What does the compiler know about any reference variable at compile time (vs. run time)?

What gets printed with the following statements (each statement is executed in the order it appears). If there is a compile time error, write "Error" and assume that line is commented out when run.

```
printable = new Thing2();

System.out.println( printable.print( true ) );          ________________________________

System.out.println( printable.print() );                ________________________________

System.out.println( printable.print( "CS11SZZ" ) );     ________________________________
```