BITS Pilani
Pilani Campus

Object Oriented Programming
CS F213

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 B, NAB
Consultation: Appointment by e-mail

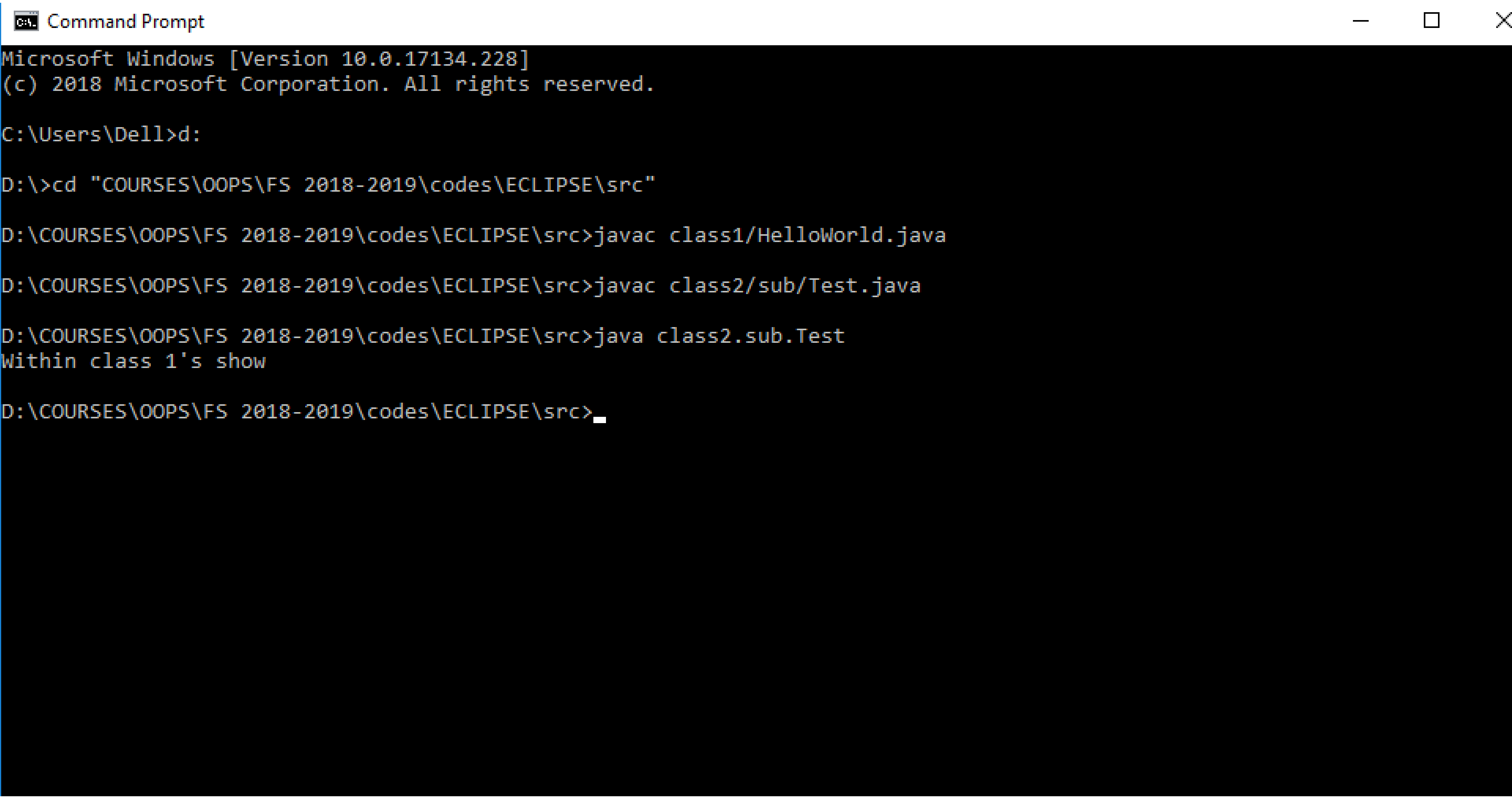
Importing a class

```
package class1;

public class HelloWorld
{
 public void show() {
  System.out.println("Within class
    1's show");
  }
}
```

```
package class2.sub;
import class1.HelloWorld;

public class Test {
public static void main(String[]
      args) {
HelloWorld h = new HelloWorld();
h.show();
}
}
```

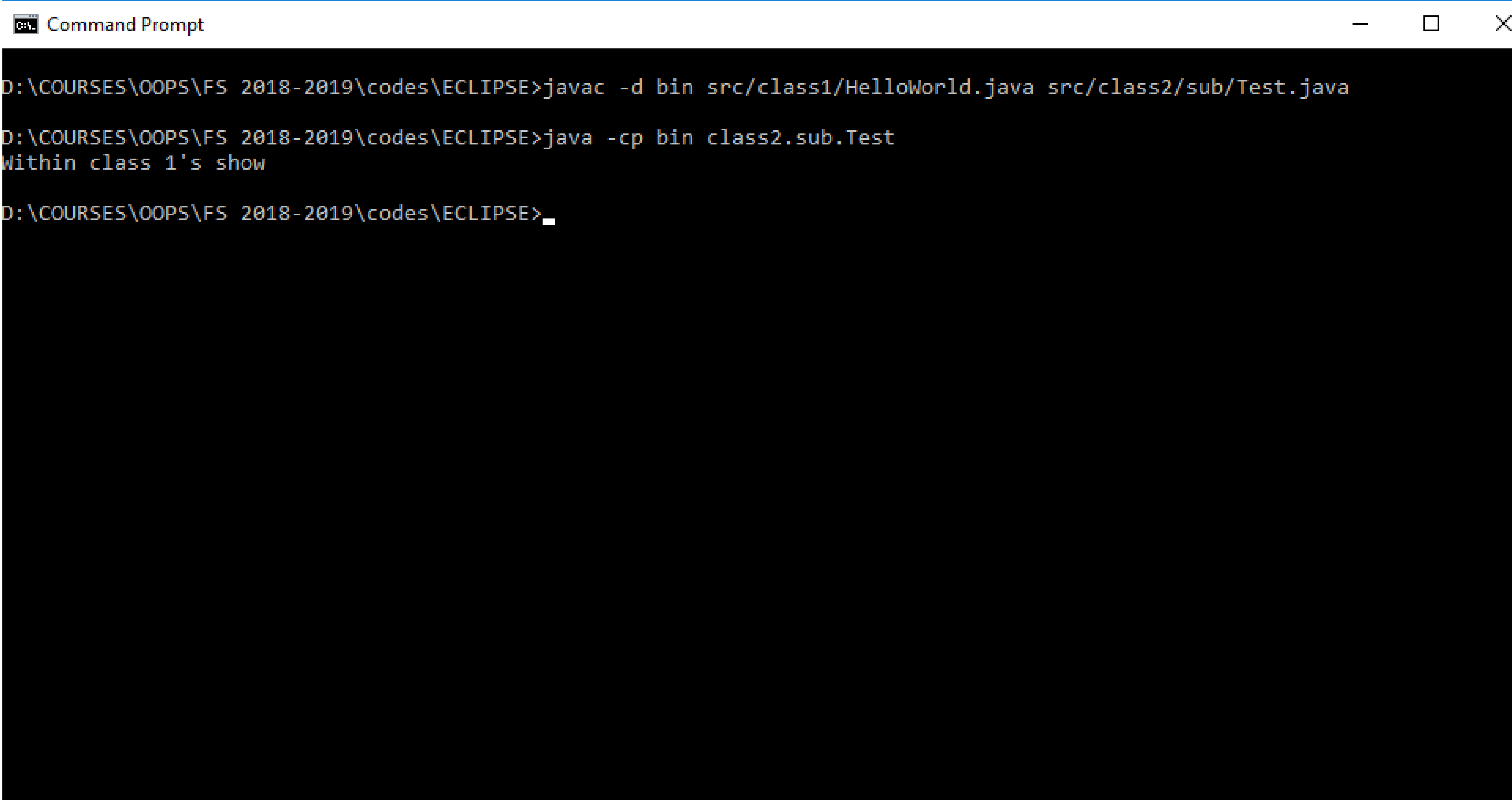
Take Home Exercise: Learn how to execute the same code from the command prompt.

Solution 1: Class files created in respective packages



```
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dell>d:

D:\>cd "COURSES\OOPS\FS 2018-2019\codes\ECLIPSE\src"

D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE\src>javac class1/HelloWorld.java

D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE\src>javac class2/sub/Test.java

D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE\src>java class2.sub.Test
Within class 1's show

D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE\src>_
```

Solution 2:Setting CLASSPATH



Command Prompt

```
D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE>javac -d bin src/class1/HelloWorld.java src/class2/sub/Test.java

D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE>java -cp bin class2.sub.Test
Within class 1's show

D:\COURSES\OOPS\FS 2018-2019\codes\ECLIPSE>_
```

innovate achieve lead

BITS Pilani

Pilani Campus

String Buffer

StringBuffer Constructors

String Buffer represents growable and writable character sequences. The size grow automatically to accommodate characters and substring insert or append operations.

Constructor	**Description**
public StringBuffer()	create an empty StringBuffer
public StringBuffer(int capacity)	create a StringBuffer with initial room for capacity number of characters
public StringBuffer(String str)	create a StringBuffer containing the characters from str

String Buffer - Methods

Methods	Description
StringBuffer append(char c)	append c to the end of the StringBuffer
StringBuffer append(int i)	convert i to characters, then append them to the end of the StringBuffer
StringBuffer append(long L)	convert L to characters, then append them to the end of the StringBuffer
StringBuffer append(float f)	convert f to characters, then append them to the end of the StringBuffer
StringBuffer append(double d)	convert d to characters, then append them to the end of the StringBuffer
StringBuffer append(String s)	append the characters in s to the end of the StringBuffer
int capacity()	return the current capacity (capacity will grow as needed).
char charAt(int index)	get the character at index.
StringBuffer delete(int start, int end)	delete characters from start to end-1
StringBuffer deleteCharAt(int index)	delete the character at index

String Buffer - Methods

Methods	Description
StringBuffer insert(int index, char c)	insert character c at index (old characters move over to make room).
StringBuffer insert(int index, String st)	insert characters from st starting at position i.
StringBuffer insert(int index, int i)	convert i to characters, then insert them starting at index.
StringBuffer insert(int index, long L)	convert L to characters, then insert them starting at index.
StringBuffer insert(int index, float f)	convert f to characters, then insert them starting at index.
StringBuffer insert(int index, double d)	convert d to characters, then insert them starting at index.
int length()	return the number of characters presently in the buffer.
StringBuffer reverse()	Reverse the order of the characters.
void setCharAt(int index, char c)	set the character at index to c.
String toString()	return a String object containing the characters in the StringBuffer.

innovate achieve lead

BITS Pilani
Pilani Campus

String Tokenizer

String Tokenizer

- **java.util.StringTokenizer** class allows you to break a string into tokens

Constructor	Description
StringTokenizer(String str)	creates StringTokenizer with specified string.
StringTokenizer(String str, String delim)	creates StringTokenizer with specified string and delimeter.
StringTokenizer(String str, String delim, boolean returnValue)	creates StringTokenizer with specified string, delimeter and returnValue. If return value is true, delimiter characters are considered to be tokens. If it is false, delimiter characters serve to separate tokens.

Methods

Public method	Description
boolean hasMoreTokens()	checks if there is more tokens available.
String nextToken()	returns the next token from the StringTokenizer object.
String nextToken(String delim)	returns the next token, after switching to the new delimiter.
boolean hasMoreElements()	same as hasMoreTokens() method.
Object nextElement()	same as nextToken() but its return type is Object.
int countTokens()	returns the total number of tokens.

String Tokenizer - Example

```
import java.util.StringTokenizer;
public class test{
 public static void main(String args[]){
   StringTokenizer st = new StringTokenizer("my name is \t khan \n");
   int i=0,j;
   j = st.countTokens();
     while (st.hasMoreTokens()) {
         System.out.println(st.nextToken());
         i++;
     }
     System.out.println("i: "+i+"and j: "+j);
     System.out.println( st.countTokens());
   }

}
```

Output:

```
my
name
is
khan
i: 4 and j: 4
0
```

String Tokenizer - Example

```
import java.util.StringTokenizer;
public class Test{
 public static void main(String args[]){
   StringTokenizer st = new StringTokenizer("my name/ is \t khan \n");
   int i=0,j;
   j = st.countTokens();
     while (st.hasMoreTokens()) {
         System.out.println(st.nextToken("/"));
         i++;
     }
     System.out.println("i: "+i+"and j: "+j);
     System.out.println( st.countTokens());
   }
}
```

Output:

```
my name
 is  khan

i: 2and j: 4
0
```

Review Questions

- StringTokenizer stuff = new StringTokenizer("abc,def,ghi");
  System.out.println(stuff.nextToken());

Output:

```
abc,def,ghi
```

- StringTokenizer stuff = new StringTokenizer("abc,def,ghi", ",");
  System.out.println(stuff.nextToken());

Output:

```
abc
```

Review Questions

- StringTokenizer stuff = new StringTokenizer("abc+def+ghi", "+", true);
  System.out.println(stuff.nextToken());
  System.out.println(stuff.nextToken());

Output:
```
abc
+
```

- StringTokenizer stuff = new StringTokenizer("abc def+ghi", "+");
  System.out.println(stuff.nextToken());
  System.out.println(stuff.nextToken());

Output:
```
abc def
ghi
```

Review Questions

- StringTokenizer st = **new StringTokenizer("abc+def:ghi", "+:", true);**
  while(st.hasMoreTokens()){
  System.***out.println(st.nextToken());*** }

Output:

```
abc
+
def
:
ghi
```

innovate achieve lead

BITS Pilani
Pilani Campus

Wrapper Classes

Wrapper Class

- Converts a primitive into object (Autoboxing) and object into a primitive (unboxing)

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

BITS Pilani
Pilani Campus

Autoboxing and Unboxing (Java 5)

Wrapper Class & Boxing - Example

```
int a = 50;
Integer i = Integer.valueOf(60);
Integer j = a;  // auto boxing
System.out.println(j.compareTo(i));
```

Output:
-1

```
Integer i = new Integer(50);
int a = i;  // unboxing
int b = Integer.numberOfLeadingZeros(i);
 System.out.println("Unboxing"+a+"Int Value"+b);
```

Output:
26

Note: Binary value of 50 is 0b110010; int is 32 bits long

Review Questions

```
static void print(int i,int j){System.out.println("int");}
static void print(Integer i,Integer j) {System.out.println("Integer");}
static void print(Integer... i){System.out.println("Var Integer");}
public static void main(String args[]){
 short s=30,t=50;
 Integer a=30,b=50,c=70;
// Place any of the following statements  }
```

Which version of the print method will be invoked?

a. **print(s)**
b. **print(s,t)**
c. **print(a,b)**
d. **print(a,b,c)**

Review Questions

```
static void print(int i,int j){System.out.println("int");}
static void print(Integer i,Integer j) {System.out.println("Integer");}
static void print(Integer... i){System.out.println("Var Integer");}
public static void main(String args[]){
 short s=30,t=50;
 Integer a=30,b=50,c=70;
// Place any of the following statements  }
```

Which version of the print method will be invoked?

a. print(s) // Error
b. print(s,t) // int
c. print(a,b) //Integer
d. print(a,b,c) //Var Integer

Rules

- **Widening and boxing cant be performed at the same time**
- **Boxing and Widening is allowed**
- **Widening > Boxing > Varargs**
- **Widening between wrapper classes is not allowed**
- **During Overloading, Widening + varargs and Boxing + varargs can only be used in a mutually exclusive manner.**