Section 10.2

10.2.1 If you redefine the Loan class in Listing 10.2 without setter methods, is the class immutable?

No. The Loan class has the getLoanDate() method that returns loanDate. loanDate is an object of the Date class. Since Date is mutable, the contents of loanDate can be changed. So, the Loan class is not immutable.

Hide Answer

Section 10.3

10.3.1 Is the BMI class defined in Listing 10.4 immutable?

Yes

Hide Answer

Section 10.4

10.4.1 What are common relationships among classes?

The common relationships among classes are association, aggregation, composition, and inheritance.

Hide Answer

10.4.2 What is association? What is aggregation? What is composition?

Association is a general binary relationship that describes an activity between two classes. Aggregation is a special form of association that represents an ownership relationship between two objects. Aggregation models has-a relationships. An object can be owned by several other aggregating objects. If an object is exclusively owned by an aggregating object, the relationship between the object and its aggregating object is referred to as a composition.

Hide Answer

10.4.3 What is UML notation of aggregation and composition?

Aggregation: empty diamond on the aggregating class. Composition: Solid diamond on the aggregating class.

Hide Answer

10.4.4 Why both aggregation and composition are together referred to as composition?

Since aggregation and composition relationships are represented using classes in the same way, we will not differentiate them and call both compositions for simplicity.

Hide Answer

Section 10.5

10.5.1 Replace the statement in line 17 in Listing 10.5 TestCourse.java so that the loop displays each student name followed by a comma except the last student name.

System.out.print(students[i] + (i < course1.getNumberOfStudents() - 1 ? ", " : " "));

Hide Answer

Section 10.7

10.7.1 Describe primitive-type wrapper classes.

Omitted

Hide Answer

10.7.2 Can each of the following statements be compiled?

```
a.      Integer i = new Integer("23");
b.      Integer i = new Integer(23);
c.      Integer i = Integer.valueOf("23");
d.      Integer i = Integer.parseInt("23", 8);
e.      Double d = new Double();
f.      Double d = Double.valueOf("23.45");
g.      int i = (Integer.valueOf("23")).intValue();
h.      double d = (Double.valueOf("23.4")).doubleValue();
i.      int i = (Double.valueOf("23.4")).intValue();
j.      String s = (Double.valueOf("23.4")).toString();
```

Hide Answer

a. Correct
b. Correct
c. Correct
d. 19
e. Incorrect, no default constructor in Double
f. Correct
g. Correct
h. Correct
i. Correct
j. Correct

10.7.3 How do you convert an integer into a string? How do you convert a numeric string into an integer? How do you convert a double number into a string? How do you convert a numeric string into a double value?

You can simply use number + "" to convert an integer to a string. Alternatively use new Integer(int).toString() to convert an integer to a string. To convert a numeric string into an integer, use Integer.parseInt(s). Use new Double(double).toString() to convert a double to a string. To convert a numeric string into a double, use Double.parseDouble(s).

Hide Answer

10.7.4 Show the output of the following code.

```
public class Test {
  public static void main(String[] args) {
    Integer x = new Integer(3);
```

```
    System.out.println(x.intValue());
    System.out.println(x.compareTo(new Integer(4)));
  }
}
```

Hide Answer

```
3
-1
```

10.7.5 What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    System.out.println(Integer.parseInt("10"));
    System.out.println(Integer.parseInt("10", 10));
    System.out.println(Integer.parseInt("10", 16));
    System.out.println(Integer.parseInt("11"));
    System.out.println(Integer.parseInt("11", 10));
    System.out.println(Integer.parseInt("11", 16));
  }
}
```

Hide Answer

```
10
10
16
11
11
17
```

Section 10.8

10.8.1 What are autoboxing and autounboxing? Are the following statements correct?

```
a.      Integer x = 3 + new Integer(5);
b.      Integer x = 3;
c.      Double x = 3;
d.      Double x = 3.0;
e.      int x = new Integer(3);
f.      int x = new Integer(3) + new Integer(4);
```

Hide Answer

a. Correct, this is same as x = new Integer(3 + 5);
b. Correct
c. Wrong, this is same as Double x = new Integer(3);
d. Correct
e. Correct
f. Correct

10.8.2 Show the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    Double x = 3.5;
    System.out.println(x.intValue());
    System.out.println(x.compareTo(4.5));
```

```
  }
}
```

Hide Answer

```
3
-1
```

Section 10.9

10.9.1 What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    java.math.BigInteger x = new java.math.BigInteger("3");
    java.math.BigInteger y = new java.math.BigInteger("7");
    java.math.BigInteger z = x.add(y);
    System.out.println("x is " + x);
    System.out.println("y is " + y);
    System.out.println("z is " + z);
  }
}
```

Hide Answer

The output is

```
x is 3
y is 7
z is 10
```

Please note that BigInteger and BigDecimal are immutable.

Section 10.10

10.10.1 Suppose that s1, s2, s3, and s4 are four strings, given as follows:

```
String s1 = "Welcome to Java";
String s2 = s1;
String s3 = new String("Welcome to Java");
String s4 = "Welcome to Java";
```

What are the results of the following expressions?

```
a.      s1 == s2
b.      s1 == s3
c.      s1 == s4
d.      s1.equals(s3)
e.      s1.equals(s4)
f.      "Welcome to Java".replace("Java", "HTML")
g.      s1.replace('o', 'T')
h.      s1.replaceAll("o", "T")
i.      s1.replaceFirst("o", "T")
j.      s1.toCharArray()
```

Hide Answer

a. true
b. false
c. true
d. true
e. true

f. Welcome to HTML
g. WelcTme tT Java
h. WelcTme tT Java
i. WelcTme to Java
j. toCharArray() returns an array of characters consisting of W, e, l, c, o, m, e, , t, o, , J, a, v, a

10.10.2 To create the string Welcome to Java, you may use a statement like this:

```
String s = "Welcome to Java";
```

or:

```
String s = new String("Welcome to Java");
```

Which one is better? Why? Hide Answer

```
String s = "Welcome to Java";
```

is better, because this type of string is stored as an interned string. The interned strings of the same value share the same object.

10.10.3 What is the output of the following code?

```
String s1 = "Welcome to Java";
String s2 = s1.replace("o", "abc");
System.out.println(s1);
System.out.println(s2);
```

Hide Answer

The output is

```
Welcome to Java
Welcabcme tabc Java
```

Hint: No method in the String class can change the content of the string. String is an immutable class.

10.10.4 Let s1 be " Welcome " and s2 be " welcome ". Write the code for the following statements:

a. Replace all occurrences of the character e with E in s1 and assign the new string to s3.

b. Split Welcome to Java and HTML into an array tokens delimited by a space and assign the first two tokens into s1 and s2.

```
a.
String s2 = s1.replaceAll('e', 'E');

b.
String[] tokens = "Welcome to Java and HTML".split(' ');
s1 = tokens[0];
s2 = tokens[1];
```

Hide Answer

10.10.5 Does any method in the String class change the contents of the string?

No

Hide Answer

10.10.6 Suppose string s is created using new String(); what is s.length()?

0

Hide Answer

10.10.7 How do you convert a char, an array of characters, or a number to a string?

Use the overloaded static valueOf method in the String class.

Hide Answer

10.10.8 Why does the following code cause a NullPointerException?

```
1  public class Test {
2    private String text;
3
4    public Test(String s) {
5      String text  = s;
6    }
7
8    public static void main(String[] args) {
9      Test test = new Test("ABC");
10     System.out.println(test.text.toLowerCase());
11   }
12 }
```

Hide Answer

The text is declared in Line 2 as a data field, but redeclared in Line 5 as a local variable. The local variable is assigned with the string passed to the constructor, but the data field is still null. In Line 10, test.text is null, which causes NullPointerException when invoking the toLowerCase() method.

10.10.9 What is wrong in the following program?

```
1  public class Test {
2    String text;
3
4    public void Test(String s) {
5      text = s;
6    }
7
8    public static void main(String[] args) {
9      Test test = new Test("ABC");
10     System.out.println(test);
11   }
12 }
```

Hide Answer

The constructor is defined incorrectly. It should not have void.

10.10.10 Show the output of the following code.

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Hi, ABC, good".matches("ABC "));
    System.out.println("Hi, ABC, good".matches(".*ABC.*"));
    System.out.println("A,B;C".replaceAll(",;", "#"));
    System.out.println("A,B;C".replaceAll("[,;]", "#"));

    String[] tokens = "A,B;C".split("[,;]");
```

```
    for (int i = 0; i < tokens.length; i++)
      System.out.print(tokens[i] +  " ");
  }
}
```

Hide Answer

false
true
A,B;C
A#B#C
A B C

10.10.11 Show the output of the following code.

```
public class Test {
  public static void main(String[] args) {
    String s = "Hi, Good Morning";
    System.out.println(m(s));
  }

  public static int m(String s) {
    int count = 0;
    for (int i = 0; i < s.length(); i++)
      if (Character.isUpperCase(s.charAt(i)))
        count++;

    return count;
  }
}
```

Hide Answer

3