Learning Targets	**Examples**

I can identify the parts of a class	0) Identify the different parts of the class below
	
	public class Planet {
	private String name;
	private double mass;
	
	public Planet() {
	name = “Earth”;
	mass = 1;
	}
	
	public Planet(String n, double m) {
	name = n;
	mass = m;
	}
	
	public String getName() {
	return name;
	}
	
	public void setMass(double m) {
	mass = m;
	}
	}

I can identify how client code connects to another class.	0.5) Identify how the different parts of the given client code connect to the Planet class above.
	
	public class ClientCode {
	public static void main(String[] args) {
	Planet a = new Planet(“Saturn”, 95.2);
	Planet b = new Planet();
	
	String x = b.getName();
	a.setMass(96);
	}
	}

I can declare instance variables for a class with the proper access permissions	1) Consider the class below:
	
	public class Teacher {
	
	}
	
	Teachers have a first name, a last name, and an integer id number. Declare instance variables to represent these pieces of information.

I can write a no-argument constructor for a class	2) Consider the class below:

	public class Bus {
	
	private int students;
	private String busDriverName;
	
	}
	
	Write a no-argument constructor for the Bus that initializes the number of students to 42 and the bus driver's name to "Lisa".

I can write a constructor that takes parameters	3) Consider the class below:
	
	public class Bus {
	
	private int students;
	private String busDriverName;
	
	}
	
	Write a two argument constructor that takes the number of students and the bus driver's name as parameters, assigning the instance variables appropriately.
	
	4) Consider the class below:
	
	public class Bus {
	
	private int students;
	private String busDriverName;
	
	}
	
	Write a one argument constructor that takes the bus driver's name as parameters. It should assign the number of students to be 0 and the bus driver's name to the parameter

I can call a constructor to create an instance of a class	5) Consider the class below:
	
	public class Desk {
	
	private int height;
	private int modelNumber;
	
	public Desk(int h, int mn) {
	height = h;
	modelNumber mn;
	}
	
	}
	
	Fill in client code so that it creates a Desk object with a height of

	30 and a model number of 9001.
	
	public class ClientCode {
	public static void main(String[] args) {
	
	}
	}
	
	6) Consider the class below:
	
	public class Desk {
	
	private int height;
	private int modelNumber;
	
	public Desk() {
	height = 20;
	modelNumber 1;
	}
	
	}
	
	Fill in client code so that it creates a Desk object.
	
	public class ClientCode {
	public static void main(String[] args) {
	
	}
	}

I can write a mutator (setter) method	7) Consider the class below:
	
	public class Bus {
	
	private int students;
	private String busDriverName;
	
	public Bus() {
	students = 0;
	busDriverName = “Lisa”;
	}
	
	}
	
	Write a mutator method that increases the number of students on the bus by a parameter.
	
	8) Change your answer to #1 so that the students are not allowed to go below 0 (in case the parameter is negative) or above 88.

I can write an accessor (getter) method	9) Consider the class below:
	
	public class Trapezoid {
	
	private int baseOne, baseTwo, height;
	
	public Trapezoid(int b1, int b2, int h) {
	baseOne = b1;
	baseTwo = b2;
	height = h;
	}
	
	}
	
	Write an accessor method that returns the length of the median of the trapezoid.
	
	10) Add another method to the Trapezoid class - this method should return the area of the trapezoid.

I can call a mutator (setter) method	11) Consider the class below:
	
	public class Student {
	
	private String firstName, lastName;
	private int pointsEarned;
	private int pointsPossible;
	
	public Student(String first, String last) {
	firstName = first;
	lastName = last;
	pointsEarned = pointsPossible = 0;
	}
	
	public void enterScore(int earned, int poss) {
	pointsEarned += earned;
	pointsPossible += poss;
	}
	
	public void changeFirstName(String first) {
	firstName = first;
	}
	
	public double getGrade() {
	return pointsEarned / (double)pointsPossible;
	}
	}
	
	Finish this client code that enters a score of 60/72 for the student.
	
	public class ClientCode {

	public static void main(String[] args) {
	Student bob = new Student(“Bob”, “Jones”);
	
	}
	}
	
	12) Now modify your answer to #1 so that we also change the Stundent’s name to “Robert”.

I can call an accessor (getter) method	13) Referring back to the Student class above, finish this client code that gets the grade of the Student and prints it as a percent.
	
	public class ClientCode {
	public static void main(String[] args) {
	Student sam = new Student(“Sam”, “Barney”);
	sam.enterScore(5, 6);
	sam.enterScore(10, 13);
	sam.enterScore(20, 21);
	
	}
	}

I can interpret the effects of the keyword **static** on variables	14) Consider the class below:
	
	public class A {
	private int b;
	private static int c;
	
	public A(int one, int two) {
	b = one;
	c = two;
	}
	
	public void increase(int one, int two) {
	b += one;
	c += two;
	}
	}
	
	What values will be stored in the two A objects after this client code executes?
	
	public class ClientCode {
	public static void main(String[] args) {
	A m = new A(5, 6);
	A n = new A(4, 2);
	
	m.increase(1, 7);
	n.increase(5, 1);
	}
	}

Answers:
#0

```
public class Planet {    //the class header
        private String name;    //private instance variable
        private double mass; //private instance variable

        public Planet() { //0-parameter constructor
                name = “Earth”;
                mass = 1;
        }

        public Planet(String n, double m) { //2-parameter constructor
                name = n;
                mass = m;
        }

        public String getName() { //instance method - specifically a getter/accessor
                return name;
        }

        public void setMass(double m) {  //instance method - specifically a setter/mutator
                mass = m;
        }
}
```

#0.5

Identify how the different parts of the given client code connect to the Planet class above.

```
public class ClientCode {
        public static void main(String[] args) {
                //calls the two parameter Planet constructor, initializing the name instance variable to “Saturn”
                //and the mass instance variable to 95.2.  The created object is stored in variable a.
                Planet a = new Planet(“Saturn”, 95.2);

                //calls the 0-parameter Planet constructor, initializing the name instance variable to “Earth”
                //and the mass instance variable to 1.  The created object is stored in variable b.
                Planet b = new Planet();

                //calls the getName() instance method, storing the result in the variable x.  The getName()
                //method has a return type of String and no parameters.  The data that will be stored in x will
                //be “Earth”
                String x = b.getName();

                //calls the setMass() instance method.  The setMass() method will have a void return type and
                //accept one integer parameter.  The mass instance variable will be changed to 96.
                a.setMass(96);
        }
}
```

#1

```
public class Teacher {

	private String firstName, lastName;
	private int id;

}
```

#2

```
public class Bus {

	private int students;
	private String busDriverName;

	public Bus() {
		students = 42;
		busDriverName = “Lisa”;
	}

}
```

#3

```
public class Bus {

	private int students;
	private String busDriverName;

	public Bus(int s, String n) {
		students = s;
		busDriverName = n;
	}

}
```

#4

```
public class Bus {

	private int students;
	private String busDriverName;

	public Bus(String n) {
		students = 0;
		busDriverName = n;
	}
```

```
}
#5

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

		Desk d = new Desk(30, 9001);

	}
}


#6

public class ClientCode {
	public static void main(String[] args) {
		Desk d = new Desk();
	}
}


#7

public class Bus {

	private int students;
	private String busDriverName;

	public Bus() {
		students = 0;
		busDriverName = “Lisa”;
	}

	public void addStudents(int num) {
		students += num;
	}

}
```

#8

```
public class Bus {

	private int students;
	private String busDriverName;

	public Bus() {
		students = 0;
		busDriverName = “Lisa”;
	}

	//Version A - don’t make any changes when exceeding limits
	public void addStudents(int num) {
		if(students + num >= 0 && students + num <= 88) {
			students += num;
		}

	}

	//Version B - stop at limits
	public void addStudents(int num) {
		students += num;
		if(students < 0) {
			students = 0;
		}
		else if(students > 88) {
			students = 88;
		}

	}
}
```

#9

```
public class Trapezoid {

	private int baseOne, baseTwo, height;

	public Trapezoid(int b1, int b2, int h) {
		baseOne = b1;
		baseTwo = b2;
		height = h;
	}

	//I chose a double return type just in case we have a situation like where baseOne is 5 and baseTwo
	//is 6
	public double getMedian() {
		double median = 0.5 * (baseOne + baseTwo);
		return median;
	}

}
```

#10

```
public class Trapezoid {

	private int baseOne, baseTwo, height;

	public Trapezoid(int b1, int b2, int h) {
		baseOne = b1;
		baseTwo = b2;
		height = h;
	}

	public double getMedian() {
		double median = 0.5 * (baseOne + baseTwo);
		return median;
	}

	//Version A
	public double getArea() {
		double area = 0.5 * (baseOne + basTwo) * height;
		return area;
	}

	//Version B - using the getMedian() function
	public double getArea() {
		return getMedian() * height;
	}

}
```

#11

```
public class ClientCode {
	public static void main(String[] args) {
		Student bob = new Student(“Bob”, “Jones”);
		bob.enterScore(60, 72);
	}
}
```

#12

```
public class ClientCode {
	public static void main(String[] args) {
		Student bob = new Student(“Bob”, “Jones”);
		bob.enterScore(60, 72);
		bob.changeFirstName(“Robert”);
	}
}
```

#13

```
public class ClientCode {
	public static void main(String[] args) {
		Student sam = new Student(“Sam”, “Barney”);
		sam.enterScore(5, 6);
		sam.enterScore(10, 13);
		sam.enterScore(20, 21);
		double gr = sam.getGrade();
		System.out.println(gr * 100 + “%”);
	} //I multiplied by 100 to change the decimal to a percent
}
```

#14

Both m and n shared the variable c (it’s static) - changing c for m also changes c for n.

Instance	m		n	
Variables	b	c	b	c
A m = new A(5, 6);	5	6	XXXXXXXXXXX	XXXXXXXXXXX
A n = new A(4, 2);	5	2	4	2
m.increase(1, 7);	6	9	4	9
n.increase(5, 1);	6	10	9	10