AP CS Unit 4: Classes and Objects Notes

An object is something that contains both data and methods. What data and what methods depend on its class. A class is generally used as a "blueprint" to create objects. In technical terms, a class _____ the data and behavior associated with some entity.

Example:

Server	**Client**

public class Square {
private int side;
 	
public Square(int x) {
side = x;
}
 	
public int area() {
int a = side*side;
return a;
}
 	
public void set(int x){
side = x;
}
}
	public class Runner {
	public static void main(String[] args) {
	
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	_____
	}
	}

A class consists of:

Instance variables: _____

_____

Constructors _____

_____

_____

An object is an _____ of a class. A constructor is used to _____ an object.

Methods _____

_____

Visibility Modifiers

public _____

private _____

Methods have three parts: a _____, _____, and _____.
Here are the two methods of the Square class.

public int area() {
  int a = side*side;
  return a;
}
	public void set(int x){
	  side = x;
	}

A **return statement** causes the program to exit the method. Let's consider three examples:

If we call method1 and n equals 3, what does the method return?
 	
If we call method1 and n equals -22, what does the method return?
	public double method1(int n) {
	  if (n < 0) {
	  return 6;
	
	  int b = 4*n;
	  return b;
	}

If we call method2 and the parameters are 3 and 5, what is displayed?
 	
Notice that this return statement (1) returns nothing and (2) causes the program to exit early.
	public void method2(int a, int b) {
	  int c = a + b;
	  if (c % 2 == 0) {
	  System.out.println("hey");
	  return;
	  }
	
	  System.out.println("joe");
	}

method3 causes this compiler error:
  unreachable statement
 	
What does this mean?
	public int method3(int c) {
	  c = c + 2;
	  return c;
	  c = c + 3;
	}

In the context of a class, all variables fall into one of three categories:

- instance variables
- parameters
- local variables

Name any instance variables
 	
Name any parameters
 	
Name any local variables
	public void method4(int h) {
	  int g = h + k;
	}

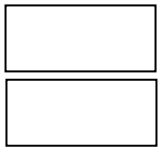
Only one line generates the compiler error:
  variable ____ might not have been initialized
 	
Which line is it?
	public void method5(int n) { // 1
	  int a; // 2
	  System.out.println(n); // 3
	  System.out.println(a); // 4
	}

Two Common Terms. If a method changes the value of an instance variable, then it is called a _____ or _____ method. If a method returns the value of an instance variable, then it is called an _____ or _____ method.

The Difference between Primitive and Object (aka Reference) Data Types

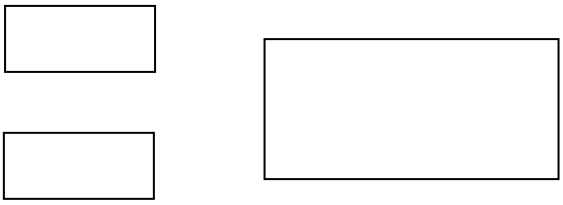
Types. First, remember the definition of a variable. A variable is a _____ _____. So, an obvious question is: what is stored in a particular variable? For primitive data types (e.g. _____ and _____) the answer is easy. The variable stores the data.

```
int x = 78;
int y = x;
```



For object variables, the answer is more complicated. An object variable contains a _____ to an object - not the object itself.

```
Point p1 = new Point( 3, 4 );
Point p2;
p2 = p1;
```



1. How many Point objects does this code create?	Point x1; Point x2;
2. How many Point objects does this code create?	Point x3 = new Point(17, -22); Point x4 = x3;

_____ is a special value that can assigned to any object variable. This indicates that the variable no longer contains a reference to an object. For example:

```
Point p1 = null;
```

If an object is instantiated but at some later time no variable contains a reference to that object, the JVM will delete that object in a process called _____. For example:

```
Point man = new Point( -1, 0 );
man = null;          // _____
```

If you attempt to call a method but the object variable does not contain a reference to an object, then you will get this runtime error: _____. For example:

```
Point pt = null;
double d = pt.distance( 0, 5.5 );        // _____
```

Another Data Type. So far we have used two types of primitive variables: ints and doubles. Another data type is the boolean data type. Variables of type boolean have a value of _____ or _____. Wherever you can use a boolean expression, you may also use a boolean variable.

Example 1. Here is one way a boolean variable may be used. In this context it is called a _____ because it signals when to keep playing and when to stop.

```
boolean game_on = true;
while ( game_on ){
        int n = (int) (7*Math.random()) + 2;
        System.out.println( n );
        if ( n == 4 )
                game_on = false;
}
```

Example 2. boolean is a common parameter type and return type for methods.

```
import java.util.Scanner;

public class Runner {
    public static void main(String[] args) {
        Scanner in = new Scanner( System.in );
        System.out.print( "Enter a number: " );
        double n = in.nextDouble();

        Calculator c = new Calculator();
        c.set( true );
        double x = c.squareIt( n );
        System.out.println( x );
    }
}
```

```
public class Calculator {
    private boolean on;

    public Calculator() {
        on = false;
    }

    public double squareIt( double a ) {
        if ( on )
            return a*a;
        else
            return -1;
    }

    public void set( boolean b ) {
        on = b;
    }
}
```

The Not Operator. We have already covered the AND (&&) and OR (||) operators. A third logical operator is the NOT (!) operator. Here are some examples:

1. What is displayed?

```
boolean boo = true;
boo = !boo;
System.out.println( boo );
```

2. Will the code execute if a = 4, b = 4, and c = 8?

3. Name values that will cause the loop to terminate.

```
while ( !( a == b && b == c ) ){
        // code
}
```

The String Class. A String object represents a sequence of one or more characters where a character could be a letter, digit, or punctuation mark. Each character in a string has a unique index starting at _____.

String s = "jump now"; // the *u* is at index _____, the *n* is at index _____

Commonly Used String Methods

Signature/Header	Example
int length()	int i = "mod 4/5".length(); System.out.println(i);

Signature/Header	Examples

int indexOf(String s)	String s1 = "bubbles";
	int x = s1.indexOf("b");
	int y = s1.indexOf("bb");
	System.out.println(x + ", " + y);

int indexOf(String s, int i)	String s1 = "bubbles";
	int x = s1.indexOf("b", 1);
	int y = s1.indexOf("bb", 3);
	System.out.println(x + ", " + y);

Signature/Header	Examples
String substring(int start)	String s1 = "phone"; String s2 = s1.substring(2); System.out.println(s2);
String substring(int start, int end)	String s1 = "phone"; String s2 = s1.substring(0, 2); System.out.println(s2);
Comments	

Signature/Header	Example
boolean equals(Object obj)	String s1 = "a"; String s2 = "A"; if (s1.equals(s2)) System.out.println("ok");

Signature/Header	Example
String toLowerCase()	String s1 = "4 SALE!"; s1 = s1.toLowerCase(); System.out.println(s1);
The above example prints 4 sale! This method did not change s1. It returned a reference to a new string that was assigned to s1.	

Signature/Header	Example
String toUpperCase()	String s1 = "What ?"; s1 = s1.toUpperCase(); System.out.println(s1);
The above example prints WHAT ? This method did not change s1. It returned a reference to a new string that was assigned to s1.	

Signature/Header	Example
String trim()	String s1 = " a b "; System.out.println(s1.length()); s1 = s1.trim(); System.out.println(s1.length());
The above example prints 9 and then 3.	

A Few Last Topics.

(1) The term _____ refers to the data in an object. The state of a String object refers to the _____ in the String. The String class has no mutator methods. Strings are _____. For example, toUpperCase does NOT make all the letters in a String uppercase; toUpperCase _____ _____

(2) String literals are String objects. For example:

```
int n = "hi".length();
```

(3) The primitive type *char* represents a single character. It is not on the AP exam but can be useful. For example:

```
char letter = 'a';
String word1 = letter + "ok";          // this compiles
String word2 = letter + letter;        // this does not compile
```

(4) Some useful escape characters are: \n _____, \t _____, \" _____

```
String s = "Say\n\"bye\"";
System.out.println( s );
System.out.println( s.length() );
```