AP CS Unit 5: Static Variables and Methods Notes

A static variable is a variable that is associated with a class and not any particular object of that class. Class/static variables exist as long as the program is running. They are independent of any objects that are instantiated.

A static method is a method that is associated with a class and not any particular object of that class.

The keyword static is used to indicate if a variable is a class variable or if a method is a class method.

Example

```
public class Car {
    private static int totalNum = 0;
    private int id;

    public Car() {
        totalNum++;
        id = totalNum;
    }

    public static int getTotalNum() {
        return totalNum;
    }

    public String toString() {
        return "Car " + id + " of " + totalNum;
    }
}
```

```
public class TestCars{
   public  static void main( String [] args ) {
        int n;
        n = Car.getTotalNum();
        System.out.println(n + " cars created.");

        Car ford = new Car();
        Car toyota = new Car();
        Car jag = new Car();

        n = Car.getTotalNum();
        System.out.println(n + " cars created.");

        System.out.println( toyota.toString() );
   }
}
```

Rules

- A class method can only access static variables
  Car.getTotalNum() ← whose instance variables?
- An instance method can access both static & instance vars.

You generally write class methods when the action is not specific to any particular object of the class.

discuss toString & show Monkey class both ways!

ie: ford.toString() prints fords instance vars.

toyota.toString() prints toyotas instance vars.

Wrapper Classes. Every primitive type has a corresponding "wrapper" class. These are simple classes that are used contain a primitive value.

Integer is the wrapper class for ints. Here's an example of how it can be used.

```
1  Integer x = new Integer( 7 );
2  System.out.println( x );          // or   System.out.println( x.toString() );
3  int y = 10 + x;                   // or   int y = 10 + x.intValue();
4  System.out.println( y );
5  Integer z = y + 3;                // or   Integer z = new Integer( y + 3 );
6  System.out.println( z );
```

Notice that in line 3, the intValue method is implicitly called. This is known as auto-unboxing. The idea is that the number inside the Integer object is taken out of its box/wrapper. In line 5, the Integer constructor is implicitly called in a process called auto-boxing. You will not be tested on this but it is useful to know.

Double is the wrapper class for doubles. Here's an example of how it can be used.

```
Double a = new Double( 6.2 );
System.out.println( a );          // or   System.out.println( a.toString() );
double d = 10 + a;                // or   double d = 10 + a.doubleValue();
System.out.println( d );
```

The wrapper classes often also have useful class methods and class constants. For example,

review

```
double x = some value
int n;
if ( x >= Integer.MIN_VALUE && x <= Integer.MAX_VALUE )
        n = (int) x;
else if ( x < Integer.MIN_VALUE )
        n = Integer.MIN_VALUE;
else
        n = Integer. MAX_VALUE;
```

Note. Wrapper classes are very useful when working with data structures that can work only with objects (and not with primitive data types). We will see this in unit 10.

→ arrayLists

Here is the source code for the beginning of the Math class. (The keyword *final* in the class declaration statement means that you cannot write a subclass of the Math class – a concept we haven't covered yet.) cannot be altered, like a constant field.

```
public final class Math {
        public static final double E = 2.7182818284590452354;
        public static final double PI = 3.14159265358979323846;

        private Math() {}

        ...
}
```

Why is the constructor for the Math class private? because all methods are static and we do not construct Math objects.

Complete the code so that the correct area of a circle with a radius of 6.4 units is calculated.

```
double radius = 6.4;
double area = Math.PI * Math.pow(radius, 2);
```

Note. Since java is an open-source language, all the source code is readily available. If you are curious, search the phrase "source code for java.lang.math" and select the link to grepcode.com to see the actual source code for the Math class.

Nested Loops. A loop within a loop is called a nested loop.

What is displayed?	`for ( int a = 0; a < 3; a++ ){`
	`  for ( int b = 5; b < 7; b++ ){`
	`  int num = a + b;`
	`  System.out.print( num + " " );`
	`  }`
	`  System.out.println();`
	`}`

What is displayed?	`for ( int a = 1; a < 4; a++ ){`
	`  int b = a;`
	`  while ( b < 5 ){`
	`  System.out.print( b + " " );`
	`  b++;`
	`  }`
	`  System.out.println();`
	`}`

you show:

they do:

a: 0
b: 5 6 7

printout:

AP CS Unit 5: Arrays
Notes

Arrays. An array is an object that consists of an indexed collection of similar items. An array has a single name and the items in an array are referred to in terms of their position within the array.

The items in an array are called elements or members. ☆ like a string

An item's position within an array is called its index or position. The first element in an array always has an index of 0. como una cadena.

```
int [] ray = new int[3];   // creates an array object called ray w/ 3 elements.
ray[ 0 ] = 6;                                                  // initially filled w/ zeros
ray[ 1 ] = 5;
ray[ 2 ] = 8;

int sum = 0;
for (int n = 0; n < ray.length; n++ )
        sum = sum + ray[ n ];

System.out.println( sum );
```

ray → | 6 | 5 | 8 | (0 1 2)

a field

public final length → from created length

note: n is the index to the array so be careful to only go up to length - 1.

19

Here are short cuts for declaring and initializing an array of values.

```
int [ ] values = { 98, 100, 95, 96 };
int[ ] values = new int[ ] { 1, 2, 3 };
```

We will only use this with primitive data types and Strings.

1. What is the value of k? 3	int [] nums = { 5, 8, 2 }; int k = nums.length;

2. What is displayed?
 	
0 , 5 , 10 , 15
	int [] nums = new int[4];
	for (int i = 0; i < nums.length; i++)
	nums[i] = 5*i;
	
	for (int i = 0; i < nums.length; i++)
	System.out.print(nums[i] + " ");

nums → | 0 | 5 | 10 | 15 | (0 1 2 3)

i: 0 1 2 3

Arrays as Parameters and Return Types

A method can specify an array as a parameter and/or return an array. If a method returns an array, the array is usually instantiated in the method. ★ same w/ a String.

For example:

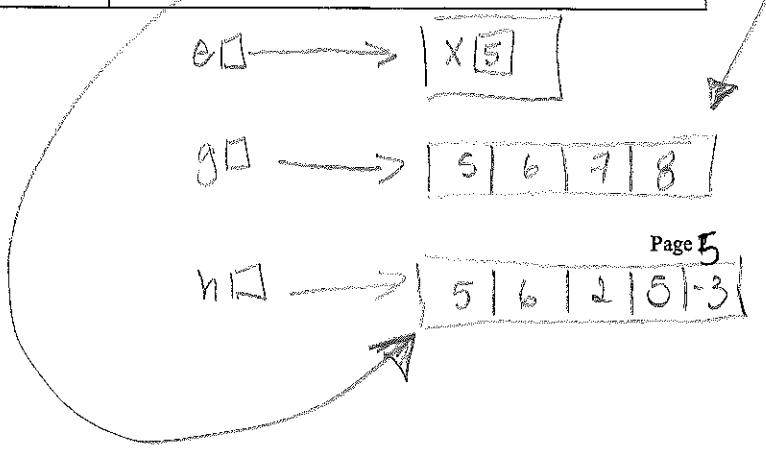
```
public class Runner {
    public static void main(String[] args) {
        Example1 e = new Example1( 5 );

        int [] g = e.method1( 4 );
        for ( int n = 0; n < g.length; n++)
            System.out.print( g[n] + " " );
        System.out.println();

        int [] h = { 5, 6, 2, 5, -3 };
        int a = e.method2( h );
        System.out.println( a );
    }
}
```

What is printed?

5 6 7 8
→ 2

```
public class Example1 {
    private int x;

    public Example1( int n ) {
        x = n;
    }

    public int [] method1( int g ){
        int [] ray = new int[ g ];
        for ( int n = 0; n < ray.length; n++ )
            ray[n] = n + x;

        return ray;
    }

    public int method2( int [] a ){
        int num = 0;
        for ( int n = 0; n < a.length; n++ ){
            if ( a[n] == x )
                num++;
        }
        return num;
    }
}
```

★ when leave ptr ★ from ray is destroyed



Common Algorithms Used with Arrays

```
public class Runner {
    public static void main(String[] args) {
        int [] ray = { 6, 5, 8, 4, 2, 5 };

        int num = Common.find1( ray );
        System.out.println( num );

        num = Common.find2( ray );
        System.out.println( num );

        num = Common.find3( ray, 5 );
        System.out.println( num );

        num = Common.find3( ray, 9 );
        System.out.println( num );
    }
}
```

What is printed?

2
30
1

```
public class Common {
    public static int find1( int [] sting ) {
        int x = sting[0];
        for ( int i = 1; i < sting.length; i++ ) {
            if ( sting[i] < x )
                x = sting[i];
        }

        return x;
    }

    public static int find2( int [] manta ) {
        int y = 0;
        for ( int i = 0; i < manta.length; i++ ) {
                y += manta[i];
        }

        return y;
    }

    public static int find3( int [] gamma, int g ) {
        for ( int i = 0; i < gamma.length; i++ ) {
                if ( gamma[i] == g )
                    return i;
        }

        return -1;
    }
}
```

What does the find1 method do? finds the smallest number in array

What does the find2 method do? sums the array

What does the find3 method do? finds the first index of int g passed into find 3 in the array passed in.

More Thoughts on Parameters and Arguments. The value of an argument is used to initialize the parameter. Changes to the value of the parameter have no effect on the argument.

```
public class Joe{
    public static void main( String[] args ){
        Music m = new Music( 11 );
        int h = 7;
        m.m1( h );
        System.out.println( h );        // ____7____

        int [] ray = { 8, 9, 13 };
        m.m2( ray );
        System.out.println( ray[0] ); // ____6____

        int [] g = { 3, 4, 5, 6 };
        m.m3( g );
        System.out.println( g[0] );     // ____3____
        System.out.println( g.length ); // ____4____

        String str = "bear";
        m.m4( str );
        System.out.println( str );      // ____bear____

        example( m );
        System.out.println( m.get() ); // ____100____
    }

    public static void example( Music x ){
        x.set( 100 );
    }
}
```

```
public class Music {
    private int x;

    public Music( int k ) {
        x = k;
    }

    public void m1( int n ) {
        n = 3;
    }

    public void m2( int [] a ) {
        a[0] = 6;
    }

    public void m3( int [] a ){
        a = new int[1];
        a[0] = 55;
    }

    public void m4( String s ){
        s = "A";
    }

    public void set( int j ){
        x = j;
    }

    public int get(){
        return x;
    }
}
```

If the parameter is a primitive data type, the method cannot change the argument.

If the parameter is a reference/object data type, then the argument passed a reference to an object and the parameter now references the same object as the argument. So the method can change the object that the argument references by calling the object's mutator methods.

If the parameter is a reference/object data type and the contents of the parameter are changed, then the object that the argument references will not be affected. if the arr is reassigned as in m4 or m3 above

And remember, arrays are objects.

Arrays of Objects

To create an array of objects, you must do two things:

(1) instantiate (create) array (not nulls, @ first)

(2) fill array w/ instantiated objects.

Suppose we want to create an array of Abc objects where the Abc class is defined as shown to the right.	public class Abc {
	private int n;
	
	public Abc(int x) {
	n = x;
	}
	
	public int method() {
	return n;
	}
	}

In a client class, we could write:

```
Abc [] list = new Abc[5];          // instantiate the object which is an array
for (int n = 0; n < list.length; n++ )
        list[n] = new ABC(n+1);    // instantiate each object that
                                   // will go in an array, looping required

int z = list[ 2 ].method();        z will hold 4
```

Another example.

```
String [] words = { "too late", "7", "What?" };
int sum = 0;
for ( int n = 0; n < words.length; n++ )      <- array
        sum += words[n].length();             <- string

System.out.println( sum );
```

What is displayed? 14

Enhanced for Loop. You may use an *enhanced for loop* if you need to get the value of every element in an array, and do NOT need to change the contents of the array. For example:

```
int [] nums = { 5, 2, 8, 4 };
int sum = 0;
for ( int x : nums )
        sum += x;
```

This kind of for loop always starts at index 0 and continues to the end of the array (unless a break statement is included). *x* represents each element of the array in turn. Changing *x* does not change the actual element in the array.

What is displayed?	String [] a = { "cat", "dog", "cathy" };
	int num = 0;
	for (String s : a){
	  if (s.indexOf("cat", 0) >= 0)
	  num++;
	}
	
	System.out.println(num);

This compiles but it is the wrong place to use an enhanced for loop. What is displayed?	int [] a = { 4, 5, 6 };
	for (int k : a){
	  k = 22;
	}
	
	System.out.println(a[0]);

Two Dimensional Arrays. A two dimensional array can be thought of as a table with both rows and columns. In reality, it is an _____

To instantiate a two-dimensional array:

```
int [] [] table = new int [6][7];      // 6 rows and 7 columns
int k = table.length;                  // table.length equals the number of rows, k is 6
int n = table[0].length;               // table[0] refers to the first row, n is 7
```

Because it is really an array of arrays, you can also instantiate a two-dimensional array like this:

```
int [][] m = { { 6, 7, 8 }, {9, 1, 5 }, {4, 2, 3} };
System.out.println( m[1][0] );
System.out.println( m[0][2] );
```

To sum all the values in a two-dimensional array:

```
int [][] matrix = { { 6, 7, 8 }, {9, 1, 5 }, {4, 2, 3} };
int total = 0;
for (int row = 0; row < matrix.length; row++) {
        for (int col = 0; col < matrix[row].length; col++)
                total += matrix[row][col];

}
```

We can rewrite this using enhanced for loops.

```
int [][] matrix = { { 6, 7, 8 }, {9, 1, 5 }, {4, 2, 3} };
int total = 0;
for (int [] row : matrix){
        for (int element : row )
                total += element;

}
```

The code to the right compiles and runs. What are the contents of the array after it is initialized?

What is displayed?

Note. method1 could have been written as follows:

```
int method1( int [][] a, int r ){
    int total = 0;
    for ( int c = 0; c < a[r].length; c++ )
        total += a[r][c];

    return total;
}
```

```
public class Runner{
    public static void main( String [] args ){
        int [][] mat = new int[3][2];
        int num = 1;
        for ( int r = 0; r < mat.length; r++ ){
            for ( int c = 0; c < mat[r].length; c++ ){
                mat[r][c] = num;
                num++;
            }
        }
        System.out.println( method1( mat[0] ) );
        System.out.println( method1( mat[2] ) );
        System.out.println( method2( mat, 0 ) );
        System.out.println( method2( mat, 1 ) );
    }

    private static int method1( int [] a ){
        int total = 0;
        for ( int c = 0; c < a.length; c++ )
            total += a[c];

        return total;
    }

    private static int method2( int [][] a, int c ){
        int total = 0;
        for ( int r = 0; r < a.length; r++ )
            total += a[r][c];

        return total;
    }
}
```