5

AP CS Unit 6: Static Variables and Methods Exercises

Static Methods and Static Variables.

Problems 1 to 3 assume that the mentioned code is in the main method of the PiggyWiggy class.

1. Will this code compile? yes

```
System.out.println( Pig.talk() );
```

2. Will this code compile? no

```
int w = Pig.getWeight();
System.out.println( w );
```

← not a static method

3. This code contains a syntax error.

```
System.out.println( Pig.MAX_WT() );
```

What is it? no parenthesis! not a method, a class constant.

4. If we added the following method to the Pig class, would it compile? no

```
public static String squeal() {
    return "I weigh " + wt + " pounds.";
}
```

↑ this is an instance variable. Needs an instance!

```
public class PiggyWiggy {
    public static void main(String [] args)
{
        // code referred to problems 1 to 3
    }
}
```

```
public class Pig {
    private int wt;
    public static final int MAX_WT=900;

    public Pig( int w ) {
        if ( w > MAX_WT )
            wt = MAX_WT;
        else
            wt = w;
    }

    public int getWeight() {
        return wt;
    }

    public static String talk() {
        return "oink";
    }
}
```

5. This code does not compile. Why?

need static on method header

6. Fix the problem so that we can call the gogogo method. What will be displayed?

8

```
public class Question {
        public static void main( String [] args ){
                int n = gogogo( 8.3 );
                System.out.println( n );
        }

        public int gogogo( double x ){
                return (int) ( x + 0.5 );
        }
}
```

7. A static variable can never be accessed in an instance method.	True	(False)
8. An instance variable can never be accessed in a class method.	(True)	False

9. The Fish class compiles. If we changed m5 from an instance method to a class method (by adding the keyword *static*), would it still compile?

yes. no instance variables used.

```
public class Fish{
    public int m5( int y ){
        int x = y + 7;
        return x;
    }
}
```

10. The Tuna class compiles. If we changed m6 from a class method to an instance method (by removing the keyword *static*), would it still compile?

yes. an instance method can use both variables! (instance vs static)

```
public class Tuna{
    public static int m6( int y ){
        int x = y + 7;
        return x;
    }
}
```

11. What is displayed?

```
public class Runner{
    public static void main( String [] args ) {
        Fowl f1 = new Fowl();
        Fowl f2 = new Fowl();
        f2.m2();
        f1.m2();
        f2.m2();
        f1.m3();
        f2.m3();
    }
}
```

f1.m3(); 6,11

f2.m3(); 6,12

```
public class Fowl{
    private static int a = 3;
    private int b = 10;

    public void m2(){
        a++;
        b++;
    }

    public void m3(){
        System.out.println(a + "," + b);
    }
}
```

12. After the two lines to the right are executed,

how many instance variables exist? 2

How many class variables exist? 1

```
Fowl f1 = new Fowl();
Fowl f2 = new Fowl();
```

13. Select the TRUE statement(s). (Assume the code compiles.)

a) x and y must be instance variables.
b) x and y must be class variables.
c) x and y might both be instance, class variables, or one of each.
d) x and y are local variables.

```
public int methodX() {
    return x + y;
}
```

14. Select the TRUE statement.

a) The method does not compile. It would compile if the keyword static were removed.
b) The method compiles. It would not compile if the keyword static were removed.
c) The method compiles and it would still compile if the keyword static were removed.
d) The method does not compile and it would still not compile even if the keyword static were removed.

```
public class Pig {
    private int x = 5;
    private static int y = 7;

    public static void m77() {
        x++;
        y--;
    }
}
```

bc instance methods can access static variables and instance variables

15. Complete the close method so that it returns true if the difference between n1 and n2 is less than d. Otherwise it returns false.	public class MyDouble{
	public static boolean close(double n1, double n2, double d){
	if (Math.abs(n1-n2) < d)
	return true;
	else
	return false;
	}
	}

16. Complete the main method so that it prints GOOD if num1 and num2 are less than 0.1 units apart. Otherwise print BAD.
 	
Call the close method defined in problem 15.
	public class Runner {
	public static void main(String[] args) {
	double num1 = Math.random();
	double num2 = Math.random();
	if (MyDouble.close(num1, num2, 0.1)
	System.out.print("GOOD");
	else
	System.out.print("Bad");
	
	}
	}

17. This code does not compile. Fix it but main method must still call the doThis method.
 	
needs static
	public class Runner {
	public static void main(String[] args) {
	doThis();
	}
	static
	public void doThis(){
	System.out.print("Hey");
	}
	}

18. The Integer class has the following method: public static int parseInt(String s)

Given a String, the parseInt method returns the corresponding int value. For example, if *s* equals "43" then the method returns 43. If the argument cannot be converted to an int then a runtime error occurs. Complete the code snippet below so that the number in str is assigned to num.

```
Scanner scan = new Scanner( System.in );
String str = scan.nextLine();
int num = Integer.parseInt(str);
```

19. Within a class, a static method cannot call a non-static method.	(True)	False
20. Within a class, an instance method cannot call a static method.	True	(False)

The Door class compiles as written.

21. If the keyword static was added to the header for the getX method, would the class still compile?
yes

22. If the keyword static was added to the header for the getY method, would the class still compile?
no

```
public class Door {
        private static int x;
        private int y;

        public Door( int n ){
                x++;
                y = n;
        }

        public int getX(){
                return x;
        }

        public int getY(){
                return y;
        }
}
```

23. What does this display?

x: 0 x 2

d1 → [y: 6]

d2 → [y: -14]

2
2
6
-14

```
public class Runner {
        public static void main(String[] args) {
                Door d1 = new Door( 6 );
                Door d2 = new Door( -14 );
                System.out.println( d2.getX() );
                System.out.println( d1.getX() );
                System.out.println( d1.getY() );
                System.out.println( d2.getY() );
        }
}
```

24. Will this compile?
yes

```
public class Paper{
        public void m1(){
                m2();
        }

        public static void m2(){
        }
}
```

25. Will this compile?

26. If the method was changed to an instance method and the instance variable was changed to a class variable, would it compile? yes, static variables can be accessed in instance methods

```
public class Scissors{
        private int x;

        public static void m1(){
                x = 5;
        }
}
```

27. All static variables must be public.	True	(False)
28. Class variables are created at the start of the program and exist before any objects of that class are instantiated.	(True)	False

See stack overflow

29. Will this compile?
 	
NO
Cannot call instance method from a static method, only call using an object
	public class Rock{
	public void m1(){
	}
	
	public static void m2(){
	m1();
	}
	}

Go to http://docs.oracle.com/javase/7/docs/api/ to answer questions 30 to 35.

30. Look up the bitCount method.
 	
If x has a value of 11, what is n?
 	
If x has a value of 16, what is n?
1
	public class Runner{
	public static void main(String [] args) {
	int x = (int)(17*Math.random());
	
	int n = Integer.bitCount(x);
	System.out.println(x + ", " + n);
	}
	}

31. Do these lines compile? no, incompatible types	int n = "4"; System.out.println(n);
32. Do these lines compile?	int n = Integer.parseInt("4"); System.out.println(n);
33. Do these lines compile? Yes but it generates the following run-time error: java.lang.NumberFormatException ✓	int n = Integer.parseInt("4,000"); System.out.println(n);
34. Do these lines compile? yes	int n = Double.SIZE; System.out.println(n); 64
35. How many class/static methods does the Double class have? 11	

36. What is displayed?
5 * * * * *
6 * * * * *
7 * * * * *
8 * * * * *
	for (int x = 5; x <= 8; x++) {
	for (int y = 0; y <= 4; y++)
	System.out.print("*");
	System.out.println();
	}

37. What is displayed?
a
1 2 3
2 3 4
3 4 5
	for (int a = 1; a <= 3; a++) {
	for (int b = 1; b <= 2; b++)
	System.out.print((a+b) + " ");
	System.out.println();
	}

38. What is displayed?
a
1 1
2 1 2
3 1 2 3
4 1 2 3 4
5 1 2 3 4 5
	for (int a = 1; a <= 5; a++) {
	for (int b = 1; b <= a; b++)
	System.out.print(b + " ");
	System.out.println();
	}

39. What is displayed?	for (int g = 6; g > 2; g--) {
	  for (int h = 1; h <= 6; h += 2)
	  System.out.print(g);
	  System.out.println();
	}

6 6 6 5 5 5 4 4 4 3 3 3 2 2 2	

40. What is displayed?	for (int a = 10; a <= 50; a += 10) {
	  int b = 2;
	  while (b < a) {
	  System.out.print(b + " ");
	  b = 2 * b;
	  }
	  System.out.println();
	}

2 4 8 2 4 8 16 2 4 8 16 2 4 8 16 32 2 4 8 16 32	

41. Complete this method. If *n* is 7, it should print the pattern shown in the top figure. If *n* is 4, it should display something like the bottom figure. Assume that n is greater than zero. You will need to use a loop within a loop.

public void m36(int n) {

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        if (i == j)
            System.out.print("0");
        else
            System.out.print("*");
    } // inner for
    System.out.println();
} // outer for
```

```
0 * * * * * *
* 0 * * * * *
* * 0 * * * *
* * * 0 * * *
* * * * 0 * *
* * * * * 0 *
* * * * * * 0
```

```
0 * * *
* 0 * *
* * 0 *
* * * 0
```