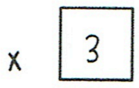
Reference Semantics

Complete the questions below with the aid of your textbook, notebook, online or classroom resources you choose. You may work on these problems with a partner, but all of your answers must be your own. The first question has been answered for you as an example.

Example:

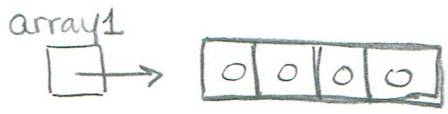
1. Write sample code and draw a picture of what memory looks like when you declare an integer variable.

code: `int x = 3;`



2. Write sample code and draw a picture of what memory looks like when you construct an array object.

`int[] array1= new int[4];`



variable array1 refers to the array

Arrange your answer to the next two questions in the graphic organizer on the next page:

3. What is one thing that reference semantics and value semantics have in common?

4. How do reference and value semantics differ?

TEALS

Reference Semantics

References to values are stored.

Values aren't copied independantly, but instead we copy references to the values.

More than 1 variable can refer to a single object.

Allows you to work with stored data.

Come in a variety of types.

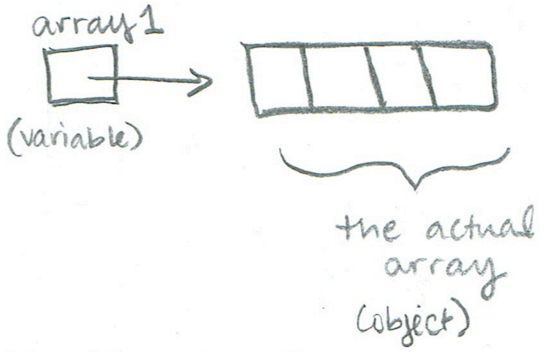
Values are stored directly.

Copy by creating independant copies of values.

Value Semantics

5. When you work with objects, are you referring to stored data or references to the data? Draw a picture to illustrate your answer.

When you work with objects you are working with references to data. Array 1 is a variable that refers to the array, not the array itself.



6. List 2 reasons that reference semantics are important to object oriented programming.

Objects like arrays can take up a lot of memory, but the references to objects are small, so using them keeps us from running out of memory.

Reference semantics lets us create multiple references to 1 object so diff. parts of our programs can all share.

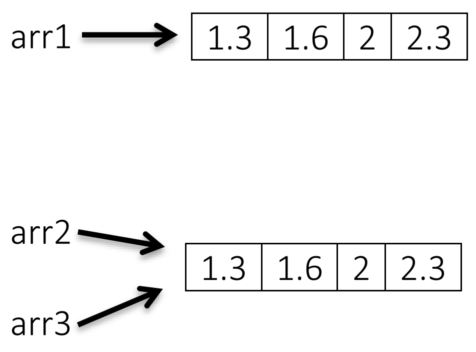
7. Why does a method to swap 2 array elements work correctly when a method to swap two integers does not? You may draw a picture to illustrate your answer, but a drawing is not required for this question.

Switching 2 integers is problematic b/c integers are value types, so there is no step where the reference to the data is seperate from its value, you have to either create copies of the values, or convert the value types to objects (reference type).

Once you have indexes to access array elements, you can pass indexes as parameters to trade places between indexes. The values exist outside of your reference to them, so extra steps such as copying aren't needed.

Reference Semantics

8. Rearrange the following code segments to create multiple references to the same object, as show in the illustration below.



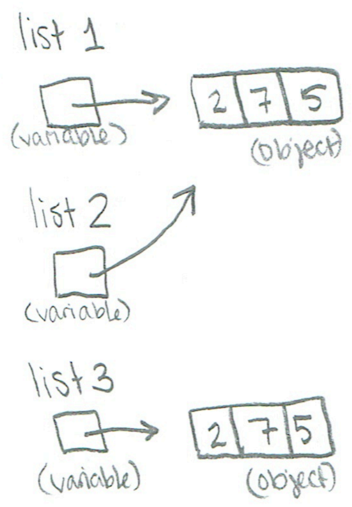
Rearrange these code segments so they execute memory symbolized by the picture above:

```
double[ ] arr3 = arr2;
}
     arr1[j] = j / 3.0 + 1;
double[ ] arr1 = new double[4];
     arr2[j] = j / 3.0 + 1;
for (int j = 0; j < arr1.length; j++){
double[ ] arr2 = new double[4];
```

Answer:

```
double[ ] arr1 = new double[4];
double[ ] arr2 = new double[4];
double[ ] arr3 = arr2;
for (int j = 0; j < arr1.length; j++){
     arr1[j] = j / 3.0 + 1;
     arr2[j] = j / 3.0 + 1;
}
```

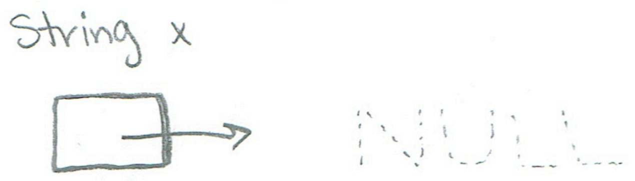
9. Draw a picture that illustrates why you can't use a == test to compare 2 objects for equivalence. Be sure to label your illustration for clarity!



- list 1 == list 2 tests if variables refer to the same object, which they do.
- If we compare list 1 & list 3, they don't refer to the same object in memory, so the test will evaluate false.
- to compare the actual arrays, we need to use the Arrays. equals or String equals methods.

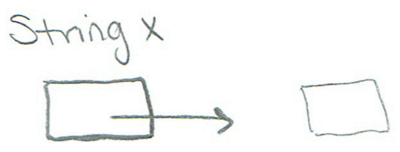
10. Draw a picture that illustrates the difference between a null object and setting a variable to an empty string. Be sure to label your illustration for clarity!

String x = null;



variable x exists, but it doesn't refer to an object yet.

String x = new String("");



variable x exists and refers to an empty array / object / string