A swap method?

• Does the following swap method work? Why or why not?

```java
public static void main(String[] args) {
    int a = 7;
    int b = 35;

    // swap a with b
    swap(a, b);

    System.out.println(a + " " + b);
}

public static void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```
• **value semantics**: Behavior where values are copied when assigned to each other or passed as parameters.
  
  – When one primitive is assigned to another, its value is copied.
  – Modifying the value of one variable does not affect others.

```java
int x = 5;
int y = x;  // x = 5, y = 5
y = 17;    // x = 5, y = 17
x = 8;     // x = 8, y = 17
```
Reference semantics

- **reference semantics**: Behavior where variables actually store the address of an object in memory.
  - When one reference variable is assigned to another, the object is *not* copied; both variables refer to the *same object*.

```java
int[] a1 = {4, 5, 2, 12, 14, 14, 9};
int[] a2 = a1;  // refers to same array as a1
a2[0] = 7;
System.out.println(a1[0]);  // 7
```

```
index: 0 1 2 3 4 5 6
value: 7 5 2 12 14 14 9
```
References and objects

• In Java, objects and arrays use reference semantics. Why?
  – *efficiency*. Copying large objects slows down a program.
  – *sharing*. It's useful to share an object's data among methods.

```java
DrawingPanel panel1 = new DrawingPanel(80, 50);
DrawingPanel panel2 = panel1;  // same window
panel2.setBackground(Color.CYAN);
```

![Diagram](image-url)
• Objects can store references to other objects as fields.
  
  Example: Homework 3 (HTML Validator)
  
  – `HtmlValidator` stores a reference to a `Queue`
  – the `Queue` stores many references to `HtmlTag` objects
  – each `HtmlTag` object stores a reference to its element `String`
Null references

- **null**: A value that does not refer to any object.
  - The elements of an array of objects are initialized to `null`.
    ```java
    String[] words = new String[5];
    ```

    ![Diagram of an array with null values](Null references)

    - not the same as the empty string " " or the string "null"
    - Why does Java have `null`? What is it used for?
Null references

- Unset reference fields of an object are initialized to `null`.

```java
public class Student {
    String name;
    int id;
}

Student timmy = new Student;
```

![Diagram](null references)
Things you can do w/ null

• store null in a variable or an array element
  String s = null;
  words[2] = null;

• print a null reference
  System.out.println(timmy.name);  // null

• ask whether a variable or array element is null
  if (timmy.name == null) { ...  // true

• pass null as a parameter to a method
  – some methods don't like null parameters and throw exceptions

• return null from a method (often to indicate failure)
  return null;
• **dereference**: To access data or methods of an object.
  
  - Done with the dot notation, such as `s.length()`.
  - When you use a `.` after an object variable, Java goes to the memory for that object and looks up the field/method requested.

```java
Student timmy = new Student();
timmy.name = "Timmah";
String s = timmy.name.toUpperCase();
```
Null pointer exception

- It is illegal to dereference `null` (it causes an exception).
  - `null` does not refer to any object, so it has no methods or data.

```java
Student timmy = new Student();
String s = timmy.name.toUpperCase(); // ERROR
```

Output:
```
Exception in thread "main"
java.lang.NullPointerException
at Example.main(Example.java:8)
```
References to same type

• What would happen if we had a class that declared one of its own type as a field?

```java
public class Strange {
    private String name;
    private Strange other;
}
```

– Will this compile?
  • If so, what is the behavior of the other field? What can it do?
  • If not, why not? What is the error and the reasoning behind it?
Linked data structures

• All of the collections we will use and implement in this course use one of the following two underlying data structures:
  
  – an **array** of all elements
    • ArrayList, Stack, HashSet, HashMap

  
  – a set of **linked objects**, each storing one element, and one or more reference(s) to other element(s)
    • LinkedList, TreeSet, TreeMap
A list node class

```java
public class ListNode {
    int data;
    ListNode next;
}
```

- Each list node object stores:
  - one piece of integer data
  - a reference to another list node

- *ListNode* objects can be "linked" into chains to store a list of values:
public class ConstructList1 {
    public static void main(String[] args) {
        ListNode list = new ListNode();
        list.data = 42;
        list.next = new ListNode();
        list.next.data = -3;
        list.next.next = new ListNode();
        list.next.next.data = 17;
        list.next.next.next = null;
        System.out.println(list.data + " " + list.next.data + " " + list.next.next.data);
        // 42 -3 17
    }
}
public class ListNode {
    int data;
    ListNode next;

    public ListNode(int data) {
        this.data = data;
        this.next = null;
    }

    public ListNode(int data, ListNode next) {
        this.data = data;
        this.next = next;
    }
}

- Exercise: Modify the previous client to use these constructors.
Linked node problem 1

- What set of statements turns this picture:

  list \[ \rightarrow \]

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

- Into this?

  list \[ \rightarrow \]

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
</tr>
</tbody>
</table>
Linked node problem 2

• What set of statements turns this picture:

list → | data | next |
       | 10   |      |

list → | data | next |
       | 20   |      |

• Into this?

list → | data | next |
       | 30   |      |

list → | data | next |
       | 10   |      |

list → | data | next |
       | 20   |      |
Linked node problem 3

• What set of statements turns this picture:

list1  →
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

list2  →
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>40</td>
</tr>
</tbody>
</table>

• Into this?

list1  →
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>30</td>
</tr>
</tbody>
</table>

list2  →
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
Linked node problem 4

• What set of statements turns this picture:

```
list    data  next
  10 → ... → 990
```

• Into this?

```
list    data  next
  10 → ... → 990 → 1000
```