Method Design

Method Design

- Once the overall class structure is decided, you need to design the class’ methods.
- At least the public methods should be described in the design.
  - They are used to interact with the rest of the system.
  - Must be described to define how that interaction happens.

Method Decomposition

- Some of the class' behaviours may be too large to reasonably implement in a single method.
- Think about splitting a task into multiple methods if…
  - … the logic is too complicated to follow easily.
  - … it's doing more than one "task".
  - … the method doesn't fit on a screen (or two).
  - … too many blocks (loops and `if`s) are nested.

Method Decomposition

- How do you decide how to split a task into multiple methods?
  - Look for distinct tasks that are logically separate.
  - … especially if they can be joined with relatively few parameters/return values being passed around.
- Look for opportunities to reuse code.
  - similar tasks that could be used in several places
  - Never copy-and-paste code.

Examples

- Assignment 1:
    - separate functions for data entry, menu, 4 stats.
    - need to pass only the array & length between them.
- Assignment 2:
    - the `swap` function in `Sorts`: in selection & quicksort
    - in `CreateArray`, `randomArray` should probably be used several times to avoid duplication of tasks.
    - Merge Sort: the merge is complicated and can be separated for readability, debugging.

Visibility

- Some of the methods created in decomposition will be specific to the implementation.
  - … and will either be useless or dangerous to call for anything but what they were designed for.
  - These methods should be declared as `private`.
  - Could be `public` if they would be useful to call by themselves.
- Helps ensure your class' state stays consistent.
  - design public interface, use private to support it.

Parameter Passing

- Note that the values passed as parameters to methods are copied when they are sent to the method.
  - … but objects are stored as references anyway. Only the reference is copied, not all of the data.
- So, the values of variables can’t be changed when passed to a function
  - … but can follow a reference and change the object it refers to.

```
public static void testParams(Student s, int num) {
    num = 20;  // change copy
    s.setFirstName("Rudiger"); // follow reference
    s = new Student(300067890, "someguy");
                               // change copy
}
public static void main(String[] args) {
    Student s1 = new Student(300012345, "userid");
    int count = 10;
    s1.setLastName("Simpson");

    testParams(s1, count);

    System.out.println(s1);
    System.out.println(count);
}
```

Output:

```
300012345: Simpson,
  Rudiger
10
```

Changing Parameters

- Bottom line: can't change the values passed to functions.
  - … but can follow references passed and change the object.
  - … if the object has methods to change it.
  - can change the local copy (parameter), but that doesn't affect outside code.