Announcements

- Today –
  - Loop Invariants
  - Finish Inheritance classwork from March 2
  - New Classwork for loop invariants
  - Turn in BOTH classworks by March 16!
- Reading Quiz for next time
- Read Chapter 7.6

Patterns

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”

- Alexander et. al, 1977
- A text on architecture!

• What is a programming or design pattern?
• Why are patterns important?
What is a pattern?

• “… a three part rule, which expresses a relation between a certain context, a problem, and a solution. The pattern is, in short, at the same time a thing, …, and the rule which tells us how to create that thing, and when we must create it.”  
  Christopher Alexander
  – name
  – problem
  – solution
  – consequences
• more a recipe than a plan, micro-architecture, frameworks, language idioms made abstract, less than a principle but more than a heuristic
• patterns capture important practice in a form that makes the practice accessible

Patterns are discovered, not invented

• You encounter the same “pattern” in developing solutions to programming or design problems
  – develop the pattern into an appropriate form that makes it accessible to others
  – fit the pattern into a language of other, related patterns

• Patterns transcend programming languages, but not (always) programming paradigms
  – OO folk started the patterns movement
  – language idioms, programming templates, programming patterns, case studies

Programming Problems

• Microsoft interview question (1998) 355788
• Dutch National Flag problem (1976)
• Remove Zeros (AP 1987) 21050084
• Quicksort partition (1961, 1986) 3891605
  31048965
• Run-length encoding (SIGCSE 1998) 1135326265353530

Loop Invariant

• What is true
  – Before the loop
  – During the loop
    – Each time before the body is started
  – After the loop
Example 1 – Find max in array

- What is the loop invariant?

```java
ArrayList<Integer> values;
...
int max = 0;
for (int k=0; k < values.size(); k++) {
    if (values.get(k) > max)
        max = values.get(k);
}
```

Example 2

- Find the max (all of them if there are several) and put them at the front of the array. The order of the other elements doesn’t matter.

- Sample input and output
  - Start: 7 3 6 9 2 8 9 9 4
  - End: 9 9 9 7 2 8 3 6 4

- Return the position of the last max

Example 2 (cont)

```java
ArrayList<Integer> values;
...
int maxPos = 0;
for (int k=0; k < values.size(); k++) {
    if (values.get(k) > values.get(maxPos))
        maxPos = k;
    else if (values.get(k) == values.get(maxPos))
        maxPos ++;
    Swap(k, maxPos);
}
```

Example 2 (cont)

```java
else if (values.get(k) == values.get(maxPos))
    {
        maxPos ++;
        Swap(k, maxPos);
    }
return maxPos;
```
**Example 3 - Removing Duplicates**

```java
void crunch(ArrayList<String> list)
{   // assume list is sorted, may have duplicates
    int lastUniqueIndex = 0;
    String lastUnique = list.get(0);
    for(int k=1; k < list.size(); k++)
    {
        String current = list.get(k);
        if (current != lastUnique)
        {
            list.set(++lastUniqueIndex, current);
            lastUnique = current;
        }
    }
    for (int k=list.size()-1; k > lastUniqueIndex; k--)
        list.remove(k);
}
```

**One loop for linear structures**

- Algorithmically, a problem may seem to call for multiple loops to match intuition on how control structures are used to program a solution to the problem, but data is stored sequentially, e.g., in an array or file. Programming based on control leads to more problems than programming based on structure. *Therefore*, use the structure of the data to guide the programmed solution: one loop for sequential data with appropriately guarded conditionals to implement the control.

Consequences: one loop really means loop according to structure, do not add loops for control: what does the code look like for run-length encoding example?

*What about efficiency?*

**Coding Pattern**

- **Name:**
  - one loop for linear structures
- **Problem:**
  - Sequential data, e.g., in an array or a file, must be processed to perform some algorithmic task. At first it may seem that multiple (nested) loops are needed, but developing such loops correctly is often hard in practice.
- **Solution:**
  - Let the structure of the data guide the coding solution. Use one loop with guarded/if statements when processing one-dimensional, linear/sequential data
- **Consequences:**
  - Code is simpler to reason about, facilitates develop of loop invariants, possibly leads to (slightly?) less efficient code

**Classwork Today**

- **APT**
  - Run Length Encoding