Functions And Abstraction

Lawrence Snyder
University of Washington, Seattle
Abstraction ... it’s all “idea”

No. 5/No. 22
Mark Rothko
When We Abstract ...

- **Abstraction** is the act of recognizing and then removing an idea or concept or process from a situation.
  - “A fox saw some juicy grapes growing on a fence. He tried and tried to reach them, but failed. Finally, he walked away, saying ‘They were probably sour’”
- Extract an idea – one failing to get something they want, often claims in the end it’s no good.
  - Abstracting – separate relevant from irrelevant
  - Recast the idea in more general terms
Recall last time ...

- We discussed Functions last time, a seemingly “obvious” idea ...
- They allow us to solve problems by first creating a useful über instruction, and then applying it to simplify our work
- Let’s recall how they work ...
The Function Becomes A Concept

- Because we noticed “process a riser,” as an action we needed to do (more than once) we think of the programming task as

- Process a riser
- Move to next riser
- Process a riser
- Move to next riser
- Process a riser
Because we noticed “process a riser,” as an action we needed to do (more than once), we think of the programming task as

- Abstracting!
- Simplifies: reduce to 5 conceptual steps rather than 21
Noticing Conceptual “Units”

- We can “see” abstractions in the problem (riser picture) or in the solution (instruction pattern) … where we find them doesn’t matter
A Five Instruction Program

Is this beautiful, or what?
Recursion Also Applies Abstraction

- A “conceptual unit” – that is, the abstraction – might apply again, immediately
Abstraction ...

- Formulating blocks of computation as a “concept” is **functional abstraction** [A better definition in a moment]
- What we did just now is important ...
  - We spotted a **coherent** (to us) part of the task
  - We solved it using a sequence of instructions
  - We put the solution into a function “package”, gave it a name, “process a riser,” and thus created a new thing, a concept, something we can talk about & use
  - Then we used it to solve something more complicated ... and then we did it again!
Collecting operations together and giving them a name is *functional abstraction*.

- The operations perform a *coherent* activity or action – they become a *concept* in our thinking.
- The operations accomplish a goal that is useful – and typically – is needed over and over again.

*Functions* implement functional abstraction: 3 parts

- A name
- A definition (instruction seq), frequently called a “body”
- Parameters – stuff inside the parentheses, covered later

```python
process_A_riser()
```
People Abstract All The Time

- Functional abstractions in which you are the agent, but someone taught you:
  - Parallel parking
  - Backstroke in swimming
- Functional abstractions you recognized and in which you are the agent
  - Doing a load of laundry
  - Making your favorite {sandwich, pizza, cookies, ...}
- Others?
No “Correct” Way To Abstract

- We have abstracted “process a riser” and “move to the next riser” as components of a solution
- As concepts, they are packaged into functions
- Maybe you thought of this in a different way
- That is, there can be other “coherent” parts of a solution

Just Do It!
The Function Is Just The Packaging

- Another way to use abstraction
If M.C. Escher handed us a problem ... what would we do?

It only simplifies our **thinking**; the bot still does all the work.
How Useful Is This Idea

- Say “Hi” to Android’s Software Stack

<table>
<thead>
<tr>
<th>Applications</th>
<th>Home</th>
<th>Contacts</th>
<th>Phone</th>
<th>Browser</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Frameworks</td>
<td>Activity Manager</td>
<td>Window Manager</td>
<td>Content Providers</td>
<td>View System</td>
<td>Notification Manager</td>
</tr>
<tr>
<td></td>
<td>Package Manager</td>
<td>Telephony Manager</td>
<td>Resource Manager</td>
<td>Location Manager</td>
<td>XMPP Service</td>
</tr>
<tr>
<td>Libraries &amp; Runtime</td>
<td>Surface Manager</td>
<td>Media Framework</td>
<td>SQLite</td>
<td>Core Libraries</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenGL/ES</td>
<td>FreeType</td>
<td>WebKit</td>
<td>Dalvik Virtual Machine (DVM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SGL</td>
<td>SSL</td>
<td>libc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>Display Driver</td>
<td>Camera Driver</td>
<td>Bluetooth Driver</td>
<td>Flash Memory Driver</td>
<td>Binder (IPC) Driver</td>
</tr>
<tr>
<td></td>
<td>USB Driver</td>
<td>Keypad Driver</td>
<td>WiFi Driver</td>
<td>Audio Drivers</td>
<td>Power Management</td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1/10/14 © 2014 Larry Snyder, CSE
And, By Request ...
Abstractions and Functions

- Abstraction is a “thinking tool” you use everyday ... in this class you will consciously apply it in programming & problem solving

- Functional Abstraction – the process of spotting a concept, “packaging” it as a function (at least in your own mind) and using it to solve some tougher problem – is ready to help when the problem is “so confusing”!