Overview

- About us: challenges & resources
- The ACM Java library: contents
- Sampling a few laboratory exercises
- Conclusions
Background

- CS 1 challenges
  - Diverse audience: varying majors and interests
  - Language choice
  - Identify interesting, authentic and accessible projects
  - What should be taught?

- Our CS 1 course
  - Lab-based: three 50-minute lectures + one 100-minute lab
  - Incorporates simple graphics
  - Uses Java and the ACM Java Libraries

- Laboratory environment
  - Linux
  - Java JDK + NetBeans
  - ACM Java Libraries → Java applets → student web sites

CCSC: MW 2008, Hope College
Course history

- AP CS language: Java
- Desire for “graphics early”
- Language evolution:
  - Pascal $\rightarrow$ C++ $\rightarrow$ Java
- Library evolution:
  - “home grown” $\rightarrow$ textbook-supplied $\rightarrow$ ACM Java libraries
- ACM Java libraries, adopted Fall 2007
How have modern programming languages affected pedagogy?*

- Complexity
  “The number of details that students must master . . . has grown much faster than the corresponding number of high-level concepts.”

- Instability
  “The languages, APIs, and tools . . . are changing more rapidly than . . . in the past.”

*http://jtf.acm.org/rationale/introduction.html
CCSC: MW 2008, Hope College
ACM Java libraries

- Released in 2006


- Libraries include:
  - Object-oriented model for programs
  - Simple methods for I/O
  - Accessible graphics capabilities

- Yield stand-alone programs or applets
Live demo: http://www.eiu.edu/~mathcs/http/
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>acm.program</code></td>
<td>Provides object-oriented program model</td>
</tr>
<tr>
<td><code>acm.io</code></td>
<td>Assists with input and output operations</td>
</tr>
<tr>
<td><code>acm.graphics</code></td>
<td>Supports simple, object-oriented graphics</td>
</tr>
<tr>
<td><code>acm.gui</code></td>
<td>Simplifies creation of interactive programs</td>
</tr>
<tr>
<td><code>acm.util</code></td>
<td>Miscellaneous utility classes</td>
</tr>
</tbody>
</table>
Input and output using the system console

ConsoleProgram: Uses a program frame with an interactive console (acm.io.Console)

DialogProgram: Input/output is performed within dialog boxes

GraphicsProgram: Fills the program frame with a GCanvas for drawing graphics objects
import acm.program.*;

public class Add extends ConsoleProgram {
    public void run() {
        int a = readInt("Enter a: ");
        int b = readInt("Enter b: ");
        println("a + b = " + (a + b));
    }
}

/*
 * File: Add2.java
 * This program ...
 */
Console program: add two integers

Enter a: 2
Enter b: 3
a + b = 5

Applet started.
import acm.program.*;

public class Add extends DialogProgram {
    public void run() {
        int a = readInt("Enter a: ");
        int b = readInt("Enter b: ");
        println("a + b = " + (a + b));
    }
}
Dialog program: add two integers

Input
Enter a:
2
OK

Input
Enter b:
3
OK

Message
a + b = 5
OK
import acm.graphics.*;
import acm.program.*;
import java.awt.*;

public class Circle extends GraphicsProgram {
    public void run() {
        int radius = 50;
        GOval myCircle = new GOval(getWidth()/2 - radius,
                                    getHeight()/2 - radius,
                                    2*radius, 2*radius);
        myCircle.setColor(Color.RED);
        myCircle.setFilled(true);
        add(myCircle);
    }
}

CCSC: MW 2008, Hope College
Graphics program: centered filled circle

CCSC: MW 2008, Hope College
Advantages

- Many details are hidden
- Unifies applets and applications
- Simple and consistent syntax
- Ability to add new functionality by manipulating or adding to underlying objects
- Entry point is run rather than main
Uses collage model; objects are added to GCanvas

Newer objects can obscure earlier ones

Distances and coordinates are measured in pixels, specified as floating point values

Several classes provided
acm.graphics: partial class diagram

java.awt.Container

GObject

GObject

GOval

GRect

GLabel

GLine

GPolygon

GImage

GArc

GCompound

GTurtle

GCanvas
Introductory lab: orientation

- Learn NetBeans IDE basics
- Download and install ACM Java libraries
- Create and modify simple Java programs
- Create several Java applets
- Use basic HTML to create a web site portfolio
import acm.graphics.*;
import acm.program.*;

public class HelloProgram extends GraphicsProgram {
    public void run() {
        add(new GLabel("Hello, world!", 100, 75));
        add(new GLabel("My Name is Julie.", 100, 95));
        add(new GLabel("I’m from Chicago.", 100, 115));
    }
}

CCSC: MW 2008, Hope College
Hello, world!
My Name is Julie.
I'm from Chicago.
Curve stitch/string art

- Recreational mathematics
- Vehicle for loops
- Parameterization: square size, number of segments
- Visual clues during debugging

CCSC: MW 2008, Hope College
Curve stitch/string art
The jailer’s problem

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>initially</strong></td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>after pass 1</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>after pass 2</td>
<td>U</td>
<td>L</td>
<td>U</td>
<td>L</td>
</tr>
<tr>
<td>after pass 3</td>
<td>U</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>after pass 4</td>
<td>U</td>
<td>L</td>
<td>L</td>
<td>U</td>
</tr>
</tbody>
</table>

- Given $n$ cells, which will end up unlocked?
- Unfamiliar problem; non-obvious pattern
- Motivates the idea of a `GNumberedRect` class
Other laboratory assignments

- Bouncing ball
- A simplified pool table
- Julia sets
- Random walk on a lattice
- ...

CCSC: MW 2008, Hope College
Conclusions

- We agree with the motivations of the ACM Java Task Force
- We found simplifications to Java helpful for CS 1
- Learning about the libraries is not a “dead-end”
- Graphics allows for interesting assignments
- Plentiful documentation and pedagogy available