

# Java in the Real World

# Final Exam Logistics

- The final exam is next **Wednesday, March 21** from 12:15PM – 3:15PM
- Rooms divvied up by last name:
  - A – B: Go to 300-300
  - C – L: Go to Cubberly Auditorium
  - M – R: Go to Hewlett 201
  - S – Z: Go to 320-105

# Final Exam Logistics

- Final is cumulative and covers chapters 1-13 of *The Art and Science of Java*.
  - Lectures on graphs and collections may be covered on the final.
  - Karel will not be covered on the final.
  - Networking and standard Java (today) will not be covered.
- Open-book, open-note, but closed-computer.

# Java in the Real World

# The ACM Libraries

- Throughout this class we've been using the ACM libraries.
  - `acm.program.*`
    - `ConsoleProgram`, `GraphicsProgram`, etc.
  - `acm.graphics.*`
    - `GOval`, `GRect`, etc.
  - `acm.util.*`
    - `RandomGenerator`
    - `ErrorException`

# The ACM Libraries

- The ACM libraries exist to simplify many common Java techniques.
- However, the ACM libraries aren't widely used outside of CS106A.
- Good news: The topics from the latter half of the quarter (file reading, arrays, **ArrayList**, **HashMap**, interactors, etc.) use only standard Java.
- We do need to cover a few last-minute details of the Java language.

“Hello, World” Without the ACM

# Starting up the Program

- In standard Java, program execution begins inside a method called  
`public static void main(String[] args)`
- The ACM libraries contain this method in the **Program** class.
- When you're not using the ACM libraries, you will have to implement this method yourself.



# Starting up the Program

In standard Java, program execution begins inside a method called

```
public static void main(String[] args)
```

The ACM libraries contain this method in the **Program** class.

When you're not using the ACM libraries, you will have to implement this method yourself.

What About Windows?

# Steps to Create a Window

- Create a new **JFrame**, which actually represents the window object.
- Add any components or interactors to the frame as you normally would.
- Set the size of the window by calling  
`frame.setSize(width, height)`
- Tell Java to quit when we close the program by calling  
`frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)`
- Show the window by calling  
`frame.setVisible(true)`

# **static** Methods

- A **static method** is a method that's specific to a *class*, rather than *instances* of that class.
- Examples:
  - `Character.isLetter`
  - `RandomGenerator.getInstance`
- Because the method is specific to the class rather than any instance, there is no receiver object.

# `public static void main`

- Because `main` is `static`, there is no instance of your class that it operates relative to.
- Common technique: Have `main` create an instance of the class and work from there.
- This is done automatically by the ACM libraries.

How are you supposed to  
remember all these methods?

**<http://docs.oracle.com/javase/7/docs/api/>**

# Graphics in the ACM Libraries

- In the ACM libraries, a program can display graphics as follows:
- Create and add a **GCanvas** component to the window.
  - **GraphicsProgram** does this automatically.
- Add the **GObjects** that need to be displayed to the canvas.



# Graphics in Standard Java

- To handle graphics in standard Java:
- Create a **JComponent** that you will use for drawing and add it to the window.
- Define a method  
`public void paintComponent(Graphics g)`  
that draws all of the graphics.
- Using the **Graphics** object, draw all the graphics you'd like!

How does **GCanvas** work?