Plan for today

• Recap: Standard Java Part 1
• Exporting Projects
• Under the hood of the ACM Libraries
Plan for today

• Recap: Standard Java Part 1
• Exporting Projects
• Under the hood of the ACM Libraries
import acm.program.*;

public class MyProgram extends ConsoleProgram {
    public void run() {
        println("Hello, world!");
    }
}

• This is a console program written using the ACM libraries.
  – It uses the ConsoleProgram class to represent a console.
  – The run method contains the program code.
  – The println method prints output to the graphical console.
• The real entry point for Java programs is the **main** method.
• The ACM Libraries implement the **main** method for us by instantiating an instance of our program and starting it running.
• **start** triggers the **init** and **run** methods to eventually execute.

```java
public static void main(String[] args) {
    new MyProgram().start(args);
}
```
Plan for today

• Recap: Standard Java Part 1
• Exporting Projects
• Under the hood of the ACM Libraries
Exporting Projects as JARs

- **JAR**: Java Archive. A compressed binary of a Java program.
  - The typical way to **distribute a Java app as a single file**.
  - Essentially just a ZIP file with Java .class files in it.

- The ACM Libraries are provided to you in a JAR file. We can also make our own runnable JAR files of our programs!
DEMO: Exporting Projects

- Note: see the handouts page of the course website for a written walkthrough of these instructions!
Plan for today

• Recap: Standard Java Part 1
• Exporting Projects
• Under the hood of the ACM Libraries
Life After The ACM Libraries

- All quarter we have relied on the **ACM Java libraries**.
  - Karel, ConsoleProgram, RandomGenerator
  - GraphicsProgram, GOval, GRect, GOval, GLine, GImage, ...

- Today we will see how we can make equivalent programs using **standard Java**.
Console Programs

• What does the `ConsoleProgram` library class do?
  – Creates a new graphical **window**
  – Puts a scrollable **text area** into it
  – Provides print and `println` commands to send text **output** to that window
  – Provides simple console output commands like `readLine` and `readInt`.
    • They display a prompt to the user, wait for input, and do error checking.
  – contains a **main method** that calls your program class's `run` method
    • `ConsoleProgram's run` is empty, but you extend and override it
Graphics Programs

The ACM library does several things to make graphics easier:

• Automatically creates and displays a **window** on the screen.
  – In standard Java, we must do this ourselves; it is called a JFrame.

• Sets up a **drawing canvas** in the center of the window
  In standard Java, we must create our own drawing canvas.

• Provides convenient methods to listen for mouse events.
  – In standard Java, event handling takes a bit more code to set up.
Let’s take some programs we can write with the ACM libraries and convert them to only use standard Java to implement their features!

- Printing to the console
- Displaying and moving around graphical text
- Getting user input
- Interactors
- Mouse events
Summary

• **Benefits of libraries:**
  – simplify syntax/rough edges of language/API
  – avoid re-writing the same code over and over
  – possible to make advanced programs quickly
  – leverage work of others

• **Drawbacks of libraries:**
  – learn a "dialect" of the language ("ACM Java" vs. "real Java")
  – lack of understanding of how lower levels or real APIs work
  – some libraries can be buggy or lack documentation
  – limitations on usage; e.g. ACM library cannot be re-distributed for commercial purposes
Summary

• Recap: Standard Java Part 1
• Exporting Projects
• Under the hood of the ACM Libraries

Next Time: Life After CS106A