Java for IOI

Java vs C++

Advantages

- Very good IDE support
- Powerful debugger and exception system
- Quick documentation
- Zero pointer shenanigans

Disadvantages

- Verbose
- Slower than C++
- Garbage collection overhead
- No/difficult low-level control

How Java works (and why you should care)

- A native program – the JVM – starts up and reads your compiled Java program

- The JVM has 2 jobs:
  - Recompile the Java bytecode into native machine code (JIT)
  - Perform maintenance tasks in the background (mostly memory management)

Java's design

- Designed for enterprise – very formal and flexible

- Everything is classes and objects

- There are no pointers

- For speed in Java:
  - Avoid creating new objects (when we really don't need to)
  - Avoid 'disposing' objects (when we really don't need to)

Strings are immutable

- Don't append to Strings in a loop – this creates a new String every time

- Use StringBuilder

```java
int numbers[] = {1, 2, 3, 4};
String output = "";

for(int num:numbers)
{
    if(!output.equals(""))
        output += " ";

    output += num;
}

System.out.println(output);
```

```java
int numbers[] = {1, 2, 3, 4};
StringBuilder output = new StringBuilder();

for(int num:numbers)
{
    if(output.length() != 0)
        output.append(' ');

    output.append(num);
}

System.out.println(output.toString());
```

Avoid the 'primitive wrappers'

- Integer, Double, etc. instead of int, double

- 1 instance per value*!

- But... generics can only use primitive wrapper classes
  - Use arrays of primitives, or make your own data structures if really necessary
  - array > ArrayList > LinkedList

I/O in Java

- Do not use Scanner!
- Use BufferedReader
- And use StringTokeniser for multiple lines
  - .split() is about 2x slower
- And then Integer.parseInt() or double.parseDouble() etc. to convert the Strings
- For output, System.out.print() works fine (buffered internally)

Other notes

- Check out the methods in the primitive classes
  - Integer, Double, Collections, Arrays
  - Various utilities for sorting, type conversions, etc.
  - Implement 'comparable' in your classes for easy sorting
- Keep everything in one file
  - Append 'static' to everything
  - Don't use packages
  - You can use inner classes
- Don't use exceptions for logic