# Programming Language Tips

Kevin Geng

8 December 2016

## 1 Java tips

### 1.1 Input/output

- Standard input and output means `System.in` and `System.out`. This is like typing in the console.

- Most of the time, `new Scanner(System.in)` should work fine for reading input. However, a `BufferedReader` and `StringTokenizer` can be faster.

- USACO requires you to read from and write to files, not standard input/output. The magical incantation in this case is:

```
BufferedReader br = new BufferedReader(new FileReader("problem.in"));
PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("problem.out")));
```

- You can reassign standard input/output using `System.setIn()` and `System.setOut()`. Alternatively, you can do this from the console: `java MyClass < input.txt > output.txt`.

- If you are using standard output, you can use standard error, or `System.err`, to print debug information. It works just like `System.out` does.

- C-style string formatting using `System.out.printf()` or `String.format()` can make things a lot more readable (though don't forget the `\n` at the end!). Compare:

```
System.out.println("Ints: " + a + ", " + b + ". Strings: " + c + ", " + d + ".");
System.out.printf("Ints: %d, %d. Strings: %s, %s.\n", a, b, c, d);
```

- It does take time to print things. If you're printing a lot of debug information in a loop, consider removing it.

### 1.2 Standard library

- For importing things, `import java.io.*; import java.util.*;` is usually enough. Your IDE may be able to automatically import classes for you.

- The `@Override` annotation makes sure you're actually overriding a superclass method.

- For arrays, the utility class `java.util.Arrays` is your friend (`binarySearch(a, key)`, `copyOf(a, newLength)`, `fill(a, val)`, `sort(a)`)! For collections, `java.util.Collections` contains all those and more.

- Some IDEs can automatically generate `equals()` and `hashCode()` for you. For arrays, use `Arrays.deepEquals` and `Arrays.deepHashCode`.

- You can create a `PriorityQueue` or `TreeMap` that uses a custom `Comparator` instead of creating a custom class that overrides `Comparable`.

- `Long.compare(a, b)` and `Double.compare(a, b)` can help you with implementing `compareTo`.

## 1.3 Gotchas

- Never use `==` unless you are comparing integers. Even then, make sure you are comparing primitives and not `Integer` objects! Instead, use `equals()`, but watch out for `NullPointerException` when you do so. It's safer to say `"hello".equals(myString)` than the other way around.

- When working with money, try to work in cents to avoid rounding issues with floating-point. Then do `String.format("%.2f", n / 100.0)` to get dollars.

- Concatenating strings in a loop will have $O(n^2)$ performance. Instead, use `StringBuilder`.

- Certain Codeforces problem writers will create test cases specially designed to make `Arrays.sort()`, which uses Quicksort, take quadratic time. If this happens, use `Integer[]` or `ArrayList<integer>`, which will use Mergesort.

# 2 Python tips

## 2.1 Input/output

- Read a line of tokens: `input().split()`. Read a line of ints: `list(map(int, input().split()))`

- Print an array separated by spaces: `print(' '.join(map(str, arr)))`

- Print to standard error, without a newline: `print('Test', file=sys.stderr, end='')`

- New-style string formatting is nice:

```
print("Ints: %d, %d. Strings: %s, %s." % (a, b, c, d))
print("Ints: {}, {}. Strings: {}, {}.".format(a, b, c, d))
```

## 2.2 Standard library

- The `itertools` module is quite excellent. In particular, see `product()`, `permutations()`, and `combinations()`, which are annoying to code otherwise.

- The `collections` module contains `defaultdict`, which sets default values for keys when you access them instead of throwing an error. Even better is `Counter`, which is a dictionary that counts the number of time an element occurs in a list. Finally, there's `collections.deque` for a stack/queue.

- There is a priority queue in the form of `heapq` which you can use tuples with... but honestly, instead of trying to figure it out, just use Java instead. (The `queue` module is designed for multithreading, so it'll be slower.)

## 2.3 Gotchas

- Python is much slower than other programming languages. Avoid it when speed matters.

- Everything above is for Python 3. If you use Python 2, don't forget about `raw_input`, integer division, and int/long, among other things. The `__future__` module can give you true division and the print function.