Chapter 5. **Formatting**

When people look under the hood, we want them to be impressed with the neatness, consistency, and attention to detail that they perceive. We want them to be struck by the orderliness. We want their eyebrows to rise as they scroll through the modules. We want them to perceive that professionals have been at work. If instead they see a scrambled mass of code that looks like it was written by a bevy of drunken sailors, then they are likely to conclude that the same inattention to detail pervades every other aspect of the project.

You should take care that your code is nicely formatted. You should choose a set of simple rules that govern the format of your code, and then you should consistently apply those rules. If you are working on a team, then the team should agree to a single set of formatting rules and all members should comply. It helps to have an automated tool that can apply those formatting rules for you.

Chapter 6. **Objects and Data Structures**

There is a reason that we keep our variables private. We don’t want anyone else to depend on them. We want to keep the freedom to change their type or implementation on a whim or an impulse. Why, then, do so many programmers automatically add getters and setters to their objects, exposing their private variables as if they were public?

**Data Abstraction**

Consider the difference between Listing 6-1 and Listing 6-2. Both represent the data of a point on the Cartesian plane. And yet one exposes its implementation and the other completely hides it.

**Conclusion**

Objects expose behavior and hide data. This makes it easy to add new kinds of objects without changing existing behaviors. It also makes it hard to add new behaviors to existing objects. Data structures expose data and have no significant behavior. This makes it easy to add new behaviors to existing data structures but makes it hard to add new data structures to existing functions. In any given system we will sometimes want the flexibility to add new data types, and so we prefer objects for that part of the system. Other times we will want the flexibility to add new behaviors, and so in that part of the system we prefer data types and procedures. Good software developers understand these issues without prejudice and choose the approach that is best for the job at hand.