CSE 1030 course notes

Abstraction and object-oriented programming

* *Abstraction* means separating tasks into levels so that, e.g., you don’t need to know anything about transistors in order to use a computer. Other system designers have created tools that we can use to interact with the computer in a simpler way.

Computer user

Uses a program (e.g. Firefox)

Provided by the application programmer

Application programmer uses the API

Designed by the system programmer

System programmer uses a system specification

System specification is designed by the system architect

The way in which this separation is enforced is called “object-oriented” programming

Data is separated into “classes” and interaction with the data is only through a specified interface

Java is an object-oriented language and everything is done through classes

Non-object-oriented languages (e.g. C)

Object-oriented languages where not everything must be a class (e.g. C++, Python)

Classes come with a public interface (API) – Application Progamming Interface

The API specifies the behavior of the class VERY PRECISELY

To implement an API, it must do EXACTLY what the API says and NOTHING ELSE.

APIs are specified for two reasons:

1. So that the object behaves in predictable ways
2. To hide the details so that the class can be updated later

Precise controls on abstraction remove a programmer’s temptation to violate abstraction – make large programs written by large teams easier to manage

Implementing APIs – utility classes (only static features)

All classes in Java have three things:

* Field: variable, data
* Methods: small snippets of code that manipulate the fields and do other things
* Constructors (later)

Keyword final: for a field, “final” means it can’t be changed after definition (thus, it must be given a value when declared)

Access specifiers:

We will see two in this course:

* public – anyone outside your class can use this field/method/constructor
* private – field/method/constructor may only be used within the class
* protected / [no specifier]