OOP Test 8.1 definitions

Jordan Boekel ID: 101625077

Abstraction

Abstraction in programming is the idea that unnecessary information is “hidden” from the user. In the test program, we abstract away the mechanism for finding a book or game in the library – The purpose of the function is described, yes (HasResource), but the underlying implementation is hidden. In essence, we attempt to write the program in a “descriptive” way, almost like english, or mathematics. Most people know what $\sqrt{\ }$ *does,* but most would be hard-pressed to compute a nontrivial root.

These hidden implementation details include everything from the functional implementation, state data in objects, etc. Ideally, we hide everything we don’t need to actually complete the function; for a mathematics function we only need the input number, for HasResource we only need the name of the resource, and so on.

Polymorphism

Polymorphism in OOP refers to the idea that a type of data or function can have many different forms (definitions). In 8.1, LibraryResource, Game and Book are examples of polymorphism; Anywhere we might want to use a LibraryResource, we can use a game or book instead. They guarantee that they can fulfill any roll that a LibraryResource can take, so there won’t be issues down the line if a programmer calls a method defined for LibraryResource, as both Game and Book will also inherit those methods.

There a many forms of polymorphism in OOP beyond this, such as differing method implementations with the same name, so that the same “result” functionality is maintained despite differences elsewhere. eg. ToString(), which converts objects to string representations; The implementation for such a function would be different for an int object compared to, say, a Book object, if we chose to define such a function for it, perhaps to return the name of the book.