Software Engineering Design: Theory and Practice
by Carlos E. Otero
Errata – January 11, 2013

Chapter 6

Page 177:
Change from "…that are efficient in the creation of object …" to "…that are efficient in the creation of objects …"

Page 181:
In the step-by-step approach for creating abstract factory, add the following step:
6. Associate each concrete factory from step 5 with their respective products from step 3.

Page 182:
In Listing 6.1, both createKeyboard and createCpu are functions, so add open and closing parenthesis after, i.e., `virtual Keyboard* createKeyboard() = 0` and `virtual Cpu* createCpu() = 0`.

Page 191:
In the step-by-step approach for creating factory method, add the following step:
5. Associate each factory from step 4 with its respective product from step 2.

Page 192 and 193:
Code Listing 6.11 is incomplete, since it presents the case where the standard computer store carries only one product. Change code Listing 6.11 to the following:

```
// The factory method for creating computer products.
Computer* StandardComputerStore::createComputer(string type)
{
    // Pointer to a computer object.
  Computer* computer = 0;

  // Determine which computer needs to be created.
  if( type.compare("standard") == 0 ) {
    // Create the StandardComputer. Clients are responsible for cleaning
    // up the memory for the computer object. Internally, StandardComputer
    // uses StandardComputerPartsFactory to create a standard computer.
    computer = new StandardComputer;
  }
  else if( type.compare("DELL") == 0 ) {
    // Create a standard DELL Computer. Clients are responsible for cleaning
    // up the memory for the computer object. Internally, StandardComputer
    // uses StandardComputerPartsFactory to create a standard computer.
    computer = new DellComputer("DellInspiron");
  }
  else if( type.compare("MAC") == 0 ) {
    // Create a standard MAC Computer. Clients are responsible for cleaning
    // up the memory for the computer object. Internally, StandardComputer
    // uses StandardComputerPartsFactory to create a standard computer.
    computer = new MacComputer("MacBookAir");
  }
  else {
    // Computer type not supported at this store. Create a null computer object.
    computer = new NullComputer(type);
  }

  // Return the newly created computer object. Clients are responsible
  // for cleaning up the computer object.
  return computer;
}
```

On page 192, change the first sentence to "As seen, the implementation for the factory method for standard computer stores support standard generic computers as well as standard MAC and standard Dell computers (via MacComputer and DellComputer). Both MacComputer and DellComputer realize the Computer interface, similarly to the StandardComputer and AdvancedComputer presented in Figure 6.2"

Page 194:

In the first paragraph, line 5, change "a creator class (i.e., the builder) that species the …" to "a creator class (i.e., the builder) that specifies the …"

Page 197:

In the Benefits section, change the first bullet from "… process with its representation…" to "process from its representation…"