

Singleton and Bridge

9-19-2007

Opening Discussion

- Overriding equals in a class hierarchy typically breaks symmetry and transitivity.
- If two objects are equal, their hash codes have to be equal.
- You don't need to write custom memory control if you don't have pointers.
- Virtual destructors needed if you have other virtual members.
- Do you have any questions about the readings for today's quiz?

Singleton

- This is another creational pattern. This time the idea is that you have a class where you want only one single instance of it to be instantiated.
- Typically when this pattern is applied there is also a global way of getting hold of that single object. This can be done by having a static method in the class.
- Another advantage is that the static method will return a pointer/reference so it is easy to have it return a subtype.

Example

- There are many examples of this that could be used. Things like file systems or memory managers where only one should be present. In Java you have classes like the Toolkit class that can give information about the system.
- As something of a general rule you call the method that gets the singleton “instance”. Obviously this doesn't have to be followed. Also make constructors non-public to enforce the singleton aspect.

Benefits and Drawbacks

- Not only can you easily return a subtype instead of the declared type, you can also decide later there should be 2, 3, or more of that type with no alterations other than to the static method.
- It's better than having a single global because the class encapsulates the instance and doesn't let other parts of code mess it up.
- Better than a utility class because of flexibility.

Bridge

- This pattern looks somewhat similar to Adapter, but is more complex and serves a somewhat different role.
- It is supposed to decouple an abstraction from its implementation so that both can be varied independently.
- This is done by putting the abstraction and the implementation in separate inheritance hierarchies instead of having them share one.

Example

- Imagine having different window abstractions and having different windowing implementations they can exist on.
- The abstraction interface (Window) will define broad functions while the implementation interface will have very specific, primitive functions.
- Then there can be implementations of both of these interfaces that do what is required for certain situations.
- Could be used with abstract factory.

Benefits and Drawbacks

- Completely decouples the interface and implementation to the point that the implementation can be changed at runtime for a single object.
- It is easier to extend the two separate hierarchies.
- Gives better hiding of implementation details.
For example you could share implementation objects, but the client will never know.

Progress Reports

- No one agreed to talk today in advance. Is there anyone who has come up with anything significant during the last week that wants to talk about it a bit?