1 Design Patterns

Design Patters are general solutions to common object-oriented problems. The goal of design patterns is to create object-oriented software that is more flexible, maintainable and reliable.

1.1 The Factory Method Pattern

- If we program to an Implementation we get locked into a specific type. Our code then needs modification if our set of concrete type get extended.
- Problem: By using the "new" operator we are forcing ourselves to a concrete implementation. e.g. Duck duck = new MallardDuck();
- Often we end up writing code with conditional logic to determine which concrete object to create: [language=C++, caption=undesired conditional logic to handle creation of object] Duck duck if (picnic) duck = new MallardDuck(); else if (hunting) duck = new DecoyDuck(); else if (inBathTub) duck = new MallardDuck();
- Here we make run-time decisions for which class to instantiate. When we see this code we know that when requirement changes, and we want to add new types, we need to open up this code and change it. This violates the Open/Close Principle.
- Lets look at what varies, the creation and encapsulate it into a factory class.