SWEN 262
Engineering of Software Subsystems

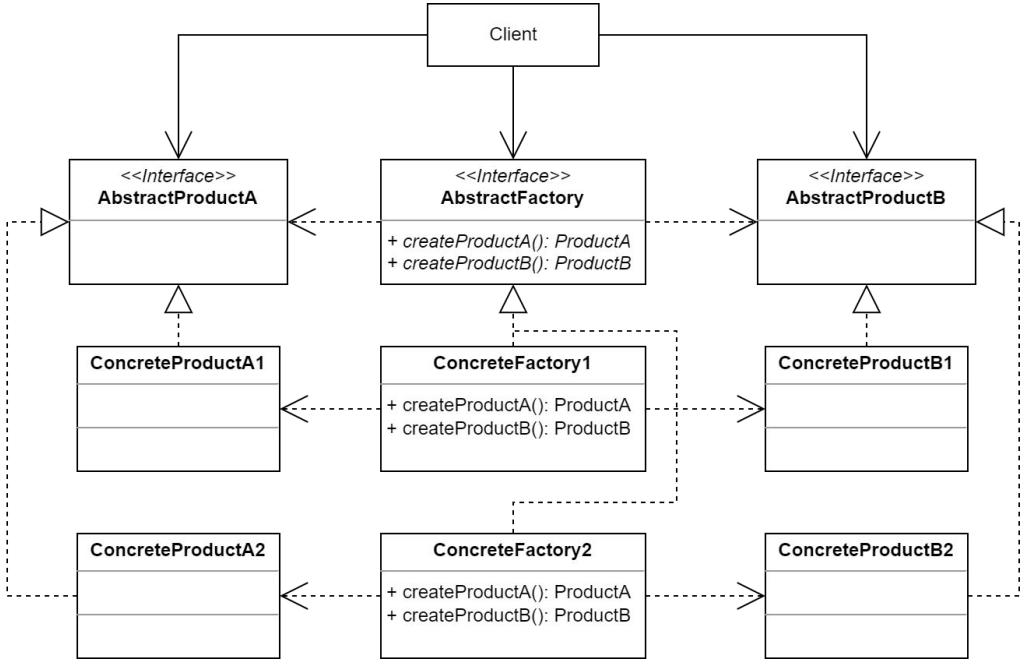
Abstract Factory Pattern

Acme Toy Complex

1. The Acme Toy Complex manufactures several different kinds of toys:
   a. *All toys have a 7-digit unique product code, a name, and a manufacturer's suggested retail price (MSRP).*
   b. *Robots walk in circles, play sounds, and can be recharged when their batteries die.*
   c. *Dolls have different hair and eye colors and play one of 10 different sounds when their string is pulled.*:
   d. *Action Figures have different hair and eye colors and play one of 10 different sounds when a button is pressed. Action figures may or may not come with Kung-Fu Grip™.*
2. When a truck arrives at the complex it is loaded with toys manufactured at the complex.
3. The product code, name, and MSRP of any toy manufactured must be kept in a log.

There are a number of different ways to design the system for tracking the toys manufactured by lots of different factories. Let's take a look at some of the options...

GoF Abstract Factory Structure



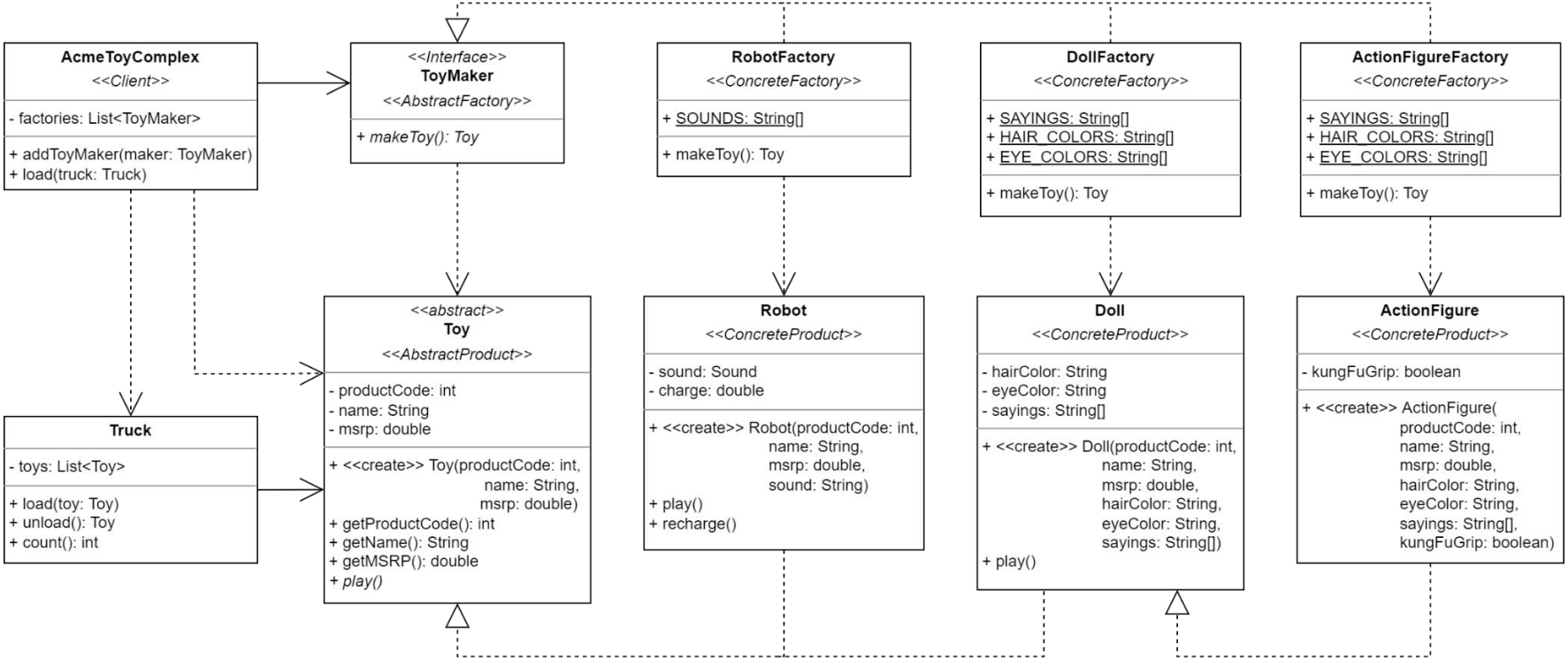
Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

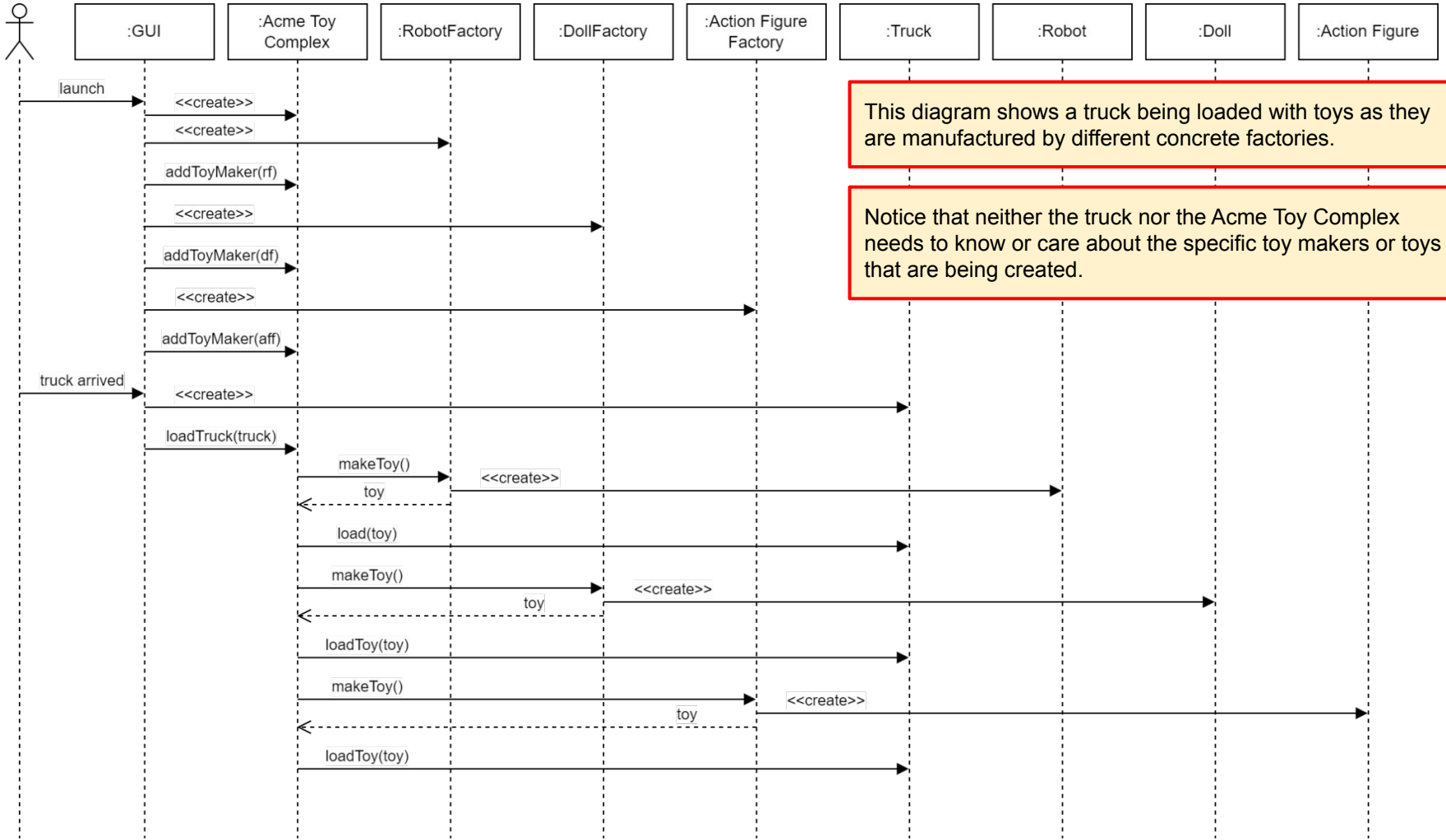
(Creational)

Toy Factory System Design

As usual, each class has a context specific name...

...but its role in the pattern is indicated in << *guillemets* >>.





This diagram shows a truck being loaded with toys as they are manufactured by different concrete factories.

Notice that neither the truck nor the Acme Toy Complex needs to know or care about the specific toy makers or toys that are being created.

GoF Pattern Card

Name: *Toy Inventory Management System*		GoF Pattern: *Abstract Factory*
Participants		
Class	Role in Pattern	Participant's Contribution in the context of the application
Toy	*Abstract Product*	*Defines the state and behavior that all toys have in common including product code, name, and manufacturer's suggested retail price (MSRP) and the play method.*
ToyFactory	*Abstract Factory*	*The interface implemented by any class that is capable of manufacturing toys of any kind. The specific toys created are determined by the implementing classes.*
Robot	*Concrete Product*	*A robot toy that can be charged to 100%. When the robot is played with, it walks in circles and plays its sound (e.g. "Bleep bloop!"). Its charge is depleted by 20% with each play. A robot can be recharged.*
Doll	*Concrete Product*	*A doll toy that has a hair color, eye color, and a pull string that will play one of 10 different sounds when it is played with.*
ActionFigure	*Concrete Product*	*An action figure toy that has hair color, eye color, and a button that will play one of 10 different sounds. Action figures may or may not come equipped with Kung-Fu Grip™.*

GoF Pattern Card

Name: *Toy Inventory Management System*		GoF Pattern: *Abstract Factory*
Participants		
Class	Role in Pattern	Participant's Contribution in the context of the application
RobotFactory	*Concrete Factory*	*A factory that builds robots. Each robot has a sound randomly selected from a set of available audio clips. Robots are at 0% charge when the are manufactured.*
DollFactory	*Concrete Factory*	*A factory that makes dolls. Each doll is given random hair and eye colors chosen from a set of available colors for both. Each doll is also given 10 sounds randomly selected from a large sound library.*
ActionFigureFactory	*Concrete Factory*	*A factory that makes action figures. Each action figure is given random hair and eye colors chosen from a set of available colors for both. Each action figure is also given 10 sounds randomly selected from a large sound library. Action figures may or may not have Kung-Fu Grip™.*
AcmeToyComplex	*Client*	*Uses the various ToyMakers to manufacture toys and load them onto trucks as they arrive. Also keeps track of the product code, name, and MSRP of any toys manufactured at the complex.*
Deviations from the standard pattern: *There is only one abstract product.*		
Requirements being covered: *1a, 1b, 1c, 1d, 2, 3*		

Abstract Factory

There are several *consequences* to implementing the abstract factory pattern:

- *Concrete classes are isolated.*
- *Exchanging between products/product families is easy.*
- *Consistency among products is promoted.*
- *Adding new kinds of products can be difficult.*

Things to Consider

1. What is the impact of abstract factory on the overall design of the system?
2. How hard would it be to add a new concrete factory that makes a new concrete product?
3. Why is adding new product types difficult? How is this similar to Visitor?