Design Issue #6

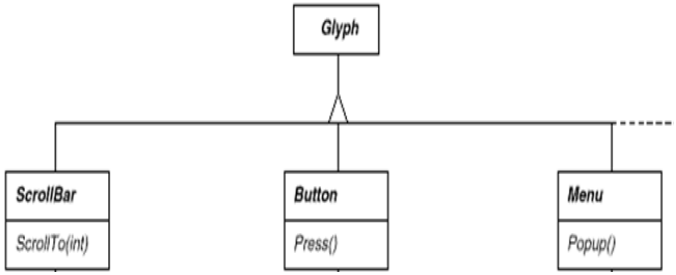
Supporting Multiple Look & Feel Standards

- One major problem in portability… consider look & feel for
  - Windows
  - Mac OS X
  - KDE
- If re-targeting is too difficult (expensive), it won't happen
- NOTE: Just one of the issues… **Look & Feel** … we deal with the Windowing system itself next

Design Issue #6

Supporting Multiple Look & Feel Standards

- Given the UI components hierarchy



- Team exercise: Build the classes to support multiple look & feel at compile / run-time, i.e. proper classes are instantiated based on the type of selected L&F.

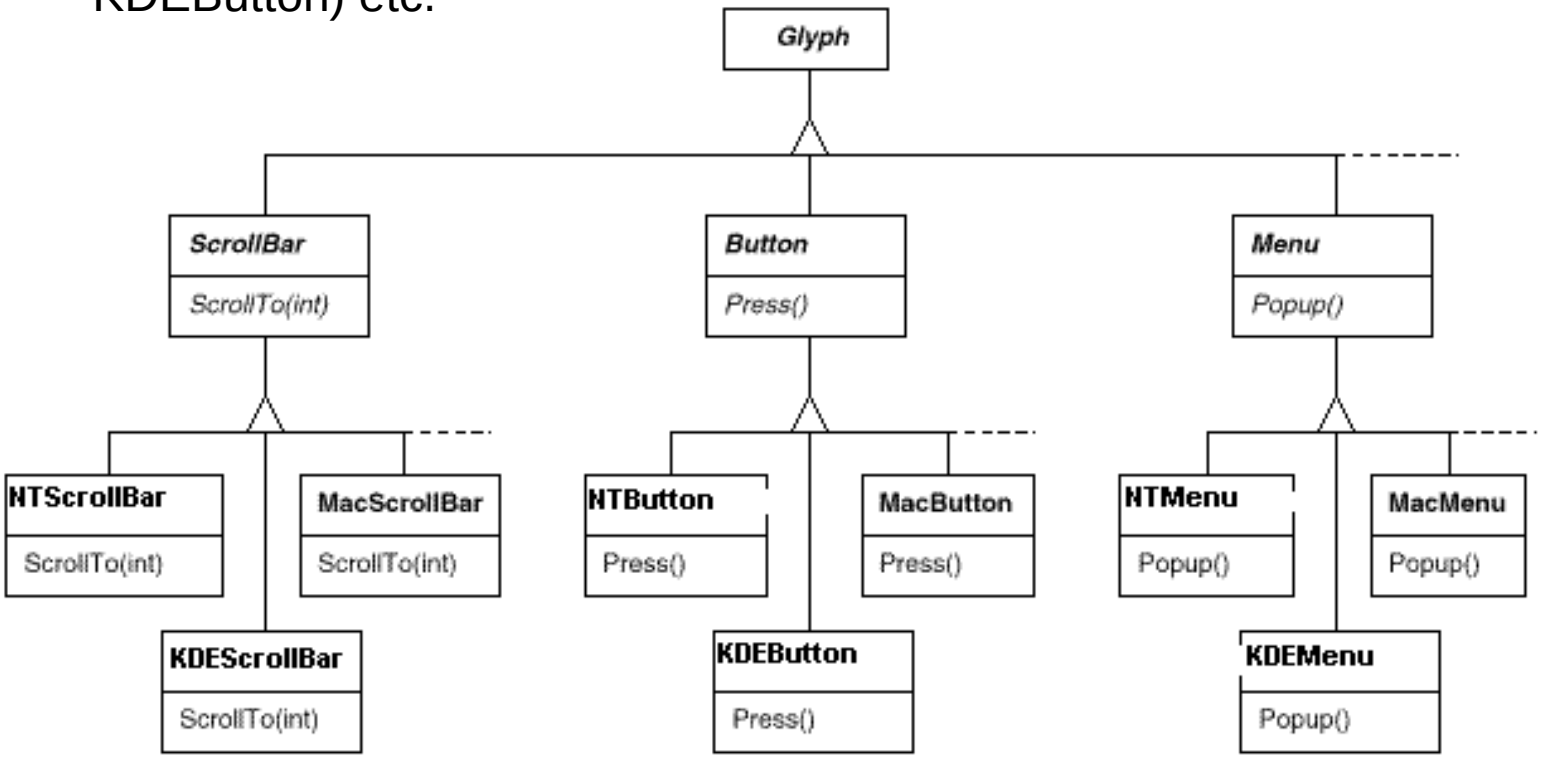
Design Issue #6

Supporting Multiple Look & Feel Standards

We use **Abstract Factory** Pattern

This allows us to define the product type at compile time / run-time (based on environment or user input)

1. First, customize the products for each platform / family (NTButton, KDEButton) etc.

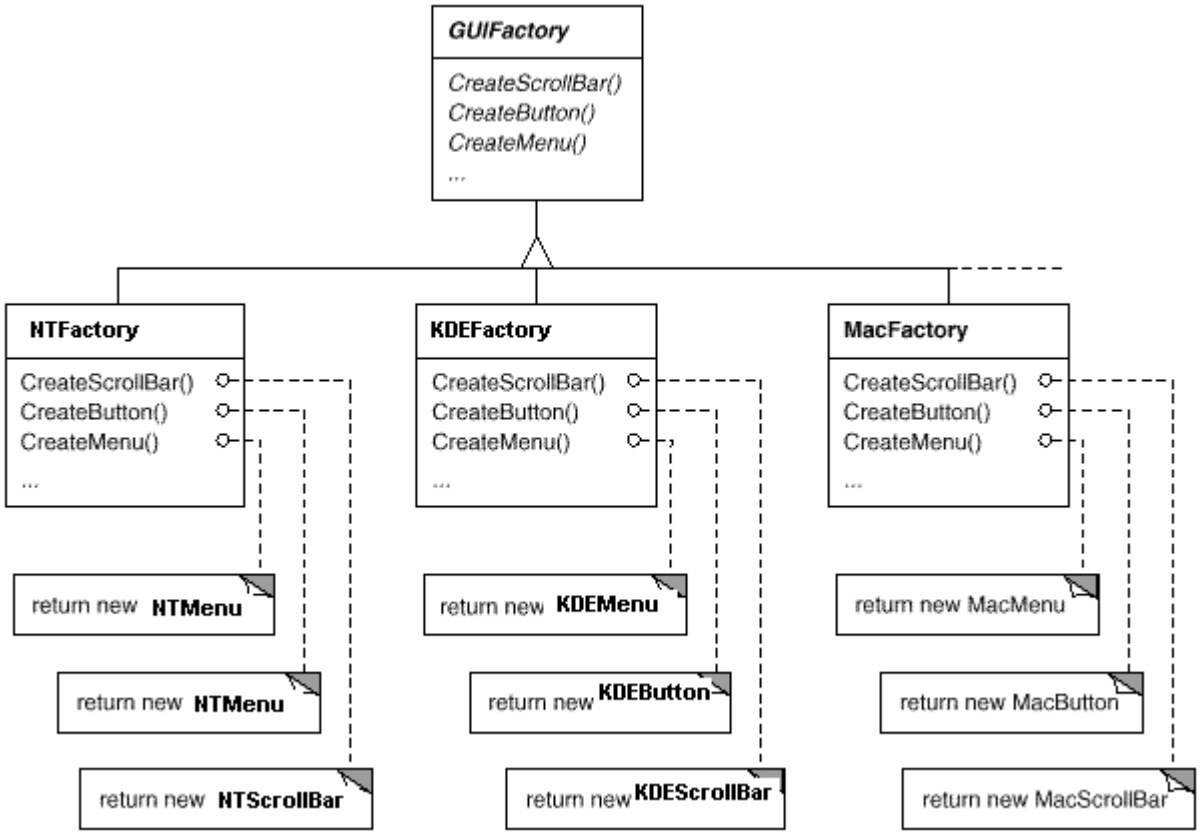


Design Issue #6

Supporting Multiple Look & Feel Standards

```
// Creating a scrollbar…
ScrollBar* sb = guiFactory->CreateScrollBar();
```

2. Second, group family instantiation in distinct class (NTFactory, KDEFactory etc)



Design Issue #6

Supporting Multiple Look & Feel Standards

- AbstractFactory classes are often implemented with
  - **Factory Method** pattern, or
  - **Prototype** pattern

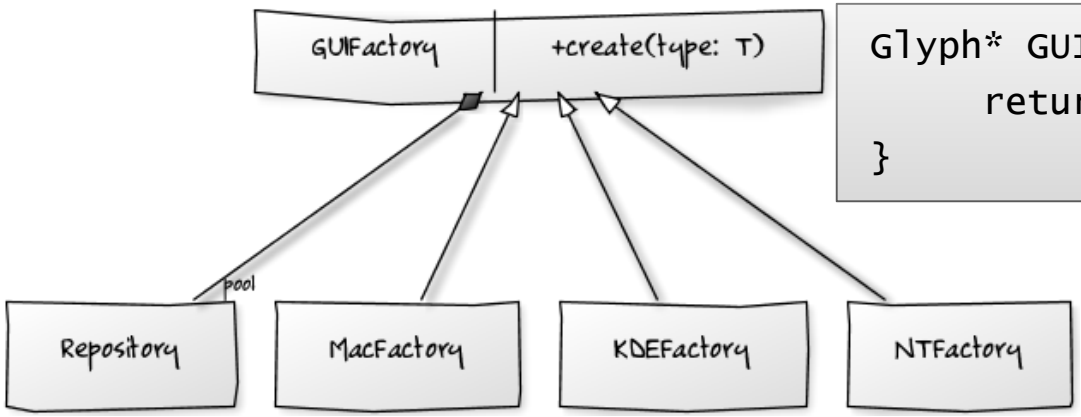
- Because a concrete factory is often unique in the system, it is implemented as a **Singleton**.

- Parameterized Factory Methods

Design Issue #6

Supporting Multiple Look & Feel Standards

Combining **Abstract Factory** and **Prototype** for a generic objects creation approach.




```
Glyph* GUIFactory::create(T type) {
      return pool->get(type)->clone();
}
```

```
MacFactory::MacFactory {
    pool = new Repository;
    pool->add(Menu::ID, new MacMenu);
    pool->add(Button::ID, new MacButton);
    pool->add(ScrollBar::ID,
              new MacScrollBar);
}
```

```
NTFactory::NTFactory {
    pool = new Repository;
    pool->add(Menu::ID, new NTMenu);
    pool->add(Button::ID, new NTButton);
    pool->add(ScrollBar::ID,
              new NTScrollBar);
}
```

```
// Creating a scrollbar…
ScrollBar* sb = guiFactory->create(ScrollBar::ID);
```