Imperative Programming II (Sheet #4)

Marius Gavrilescu

1. The Iterator design pattern could be used when a common interface for iterating over collections with different behaviours and APIs.

2. The Decorator and the Factory pattern.

3. The Decorator pattern is used to capture the different optional features of cars. Thus we have the abstract decorator Option with implementations DieselEngine, AutomaticTransmission and AlarmSystem.

4. The Factory pattern is used to capture the differences between the UK and US branches. Thus we have the abstract factory Manufacturer, the concrete factories UKManufacturer and USManufacturer which build objects of classes Convertible, Family, Sports (all being concrete implementations of Car).

5. ```
   class USManufacturer {
     def build(carType : String) : Car = {
       val car = carType match {
         case "convertible" => new Convertible;
         case "family" => new Family;
         case "sports" => new Sports;
       };
       car.temperatureScale = "Fahrenheit";
       car.driverSide = "Right";
       car
     }
   }

   class UKManufacturer {
     def build(carType : String) : Car = {
       val car = carType match {
         case "convertible" => new Convertible;
         case "family" => new Family;
         case "sports" => new Sports;
       };
       car.temperatureScale = "Celsius";
       car.driverSide = "Left";
       car
     }
   }
   ```
   ```