July 12, 2010

Review of Polymorphism

- A reference to an object can point to objects of equal type or derived from equal type.
- A reference pointing to a derived class holds these properties:
    - It may execute non-overridden methods of the reference's type.
    - It may execute the "updated" overridden methods in the derived type.
    - It may not not execute methods unique to the derived type.

The Factory Method Pattern

Vehicle.java

```
abstract class Vehicle {
	private int speed;
	private String name;

	public Vehicle(String name, int speed) {
		this.name = name;
		this.speed = speed;
	}

	final public int getSpeed() {
		return speed;
	}

	public String toString() {
		return name;
	}
}
```

Plane.java

```
public class Plane extends Vehicle {
	private int seats;
	private int engines;

	public Plane(int speed, int seats, int engines) {
		super("Plane", speed);
		this.seats = seats;
		this.engines = engines;
	}

	public int getSeats() { return seats; }
	public int getEngines() { return engines; }
```

```
}
```

Train.java

```
public class Train extends Vehicle {
	private int cars;

	public Train(int speed, int cars) {
		super("Train", speed);
		this.cars = cars;
	}

	public int getCars() { return cars; }
}
```

Factory.java

```
import java.util.*;

public class Factory {
	public static Vehicle makeVehicle(String parts) {
		Scanner scan = new Scanner(parts);

		String type = scan.next(); // Read the vehicle type

		if (type.equals("Train")) {
			int speed = scan.nextInt();
			int cars = scan.nextInt();
			return new Train(speed, cars);
		}

		if (type.equals("Plane")) {
			int speed = scan.nextInt();
			int seats = scan.nextInt();
			int engines = scan.nextInt();
			return new Plane(speed, seats, engines);
		}

		System.out.println("Unknown vehicle: "+type);
		System.exit(1);

		return new Train(0,0); // Should never be called.
	}
```

```
}
```

FactoryTest.java

```
import java.util.*;
import java.io.*;


public class FactoryTest {
	public static void main(String[] args) throws Exception {
		Scanner scan = new Scanner(System.in);


		System.out.print("Enter a filename: ");
		String filename = scan.next();


		Scanner filehandle = new Scanner(new File(filename));
		int vehicle_count = filehandle.nextInt();
		filehandle.nextLine();


		Vehicle[] vehicles = new Vehicle[vehicle_count];


		for (int i = 0; i < vehicle_count; i++) {
			String parts = filehandle.nextLine();
			vehicles[i] = Factory.makeVehicle(parts);
		}


		for (int i = 0; i < vehicle_count; i++) {
			System.out.println(vehicles[i]+" runs at speed "+vehicles[i].getSpeed());
		}
	}
}
```

parts.txt

```
4
Train 15 20
Plane 300 100 6
Plane 100 2 1
Train 30 50
```

Putting it all together

```
jcchurch@mccarthy:~/code/java/FactoryPattern$ javac FactoryTest.java
jcchurch@mccarthy:~/code/java/FactoryPattern$ java FactoryTest
Enter a filename: parts.txt
Train runs at speed 15
```

Plane runs at speed 300

Plane runs at speed 100

Train runs at speed 30