Week 6 Project—Design Patterns

**Scenario**

In this week’s Project you will implement a design pattern

**Rubric**

Point distribution for this activity:

|  |  |  |
| --- | --- | --- |
| **Activity** | | |
| **Document** | **Points possible** | **Points received** |
| Lab | 50 |  |
| **Total Points** | **50** |  |

1. List 5 design patterns commonly used in programming:

1. Prototype Method

2. Adapter Method

3. Proxy Method

4. Command Method

5. Mediator Method

2. We will create a **factory** pattern. The factory pattern is useful when you have to create objects based on user input or when you need to defer the instantiation of objects to subclasses. An object is created by calling a factory method. The factory method uses an abstract class for the superclass. Any subclasses extend the super class. Below you will have an example of a Factory Design Pattern. This is one of the most popular design patterns used in industry today. Use this example to create your own! It can be a factory of colors, shapes, animals, vehicles, toys, food, products, employees, etc. Anything you want!

Include ALL of your classes (you should have 5 at the minimum), and a screenshot of the output.

Below is an example using an abstract phone class as the superclass. The two subclasses are iphone and android. The phonefactory class provides the instantiation of the class. The factory is the main class. Notice that the Samsung and iphonex variables are of type Phone).

Phone.java – this is the super class and it is abstract

public abstract class Phone {

//The abstract methods in Phone must be implemented in the subclasses

public abstract String getRAM();

public abstract String getStorage();

public abstract String getSize();

public abstract void slogan();

@Override

public String toString(){

return "RAM= "+this.getRAM()+", Storage="+this.getStorage()+", Size="+this.getSize();

}

}

Android.java – subclass 1

public class Android extends Phone{

private String ram;

private String storage;

private String size;

public Android(String r, String s, String sz){

ram=r;

storage=s;

size=sz;

}

@Override

public String getRAM() {

return this.ram;

}

@Override

public String getStorage() {

return this.storage;

}

@Override

public String getSize() {

return this.size;

}

public void slogan()

{

System.out.println("Be together, not the same");

}

}

iPhone.java – subclass 2

public class iPhone extends Phone {

private String ram;

private String storage;

private String size;

public iPhone(String r, String s, String sz){

ram=r;

storage=s;

size=sz;

}

@Override

public String getRAM() {

return this.ram;

}

@Override

public String getStorage() {

return this.storage;

}

@Override

public String getSize() {

return this.size;

}

public void slogan()

{

System.out.println("Think different");

}

}

PhoneFactory – factory class that instantiates the object requested

public class PhoneFactory {

public Phone getPhone(String PhoneType, String r, String s, String sz)

{

//Create object here

if(PhoneType.equalsIgnoreCase("ANDROID"))

return new Android(r, s, sz);

else if(PhoneType.equalsIgnoreCase("IPHONE"))

return new iPhone(r, s, sz);

else

return null;

}

}

Main – Factory.java

public class Factory {

public static void main(String[] args) {

PhoneFactory pf = new PhoneFactory();

Phone samsung= pf.getPhone("android","6 GB","64 GB","6.0 inches");

Phone iphonex= pf.getPhone("iphone","3 GB","64 GB","5.8 inches");

System.out.println("Android Configuration:"+samsung);

samsung.slogan();

System.out.println("iPhone Configuration:"+iphonex);

iphonex.slogan();

}

}

Results of the above code:

Android Configuration:RAM= 6 GB, Storage=64 GB, Size=6.0 inches

Be together, not the same

iPhone Configuration:RAM= 3 GB, Storage=64 GB, Size=5.8 inches

Think different

BUILD SUCCESSFUL (total time: 0 seconds)

Your code:

//// Main.java START ///

public class CEIS\_420\_Norment\_Xavier\_Project\_pt1 {

public static void main(String[] args) {

CarFactory cf = new CarFactory();

Car camry = cf.getCar("toyota", "Camry", "sedan", "mid-size", "high mpg with hybrid available");

Car maverick = cf.getCar("ford", "Maverick", "pickup", "small", "hybrid available");

Car bolt = cf.getCar("chevy", "Bolt", "hatchback", "subcompact", "full electric (EV)");

Car versa = cf.getCar("nissan", "Versa", "sedan", "compact", "cheapest new car on the US open market");

System.out.println("Camry configuration: "+camry);

camry.slogan();

System.out.println("Maverick configuration: "+maverick);

maverick.slogan();

System.out.println("Bolt Configuration: "+bolt);

bolt.slogan();

System.out.println("Versa configuration: "+versa);

versa.slogan();

}

}

/// Main.java END ///

/// Car.java START ///

public abstract class Car {

public abstract String getModel();

public abstract String getStyle();

public abstract String getSize();

public abstract String getSpecial();

public abstract void slogan();

@Override

public String toString(){

return "Model is "+this.getModel()+" in "+this.getStyle()+" style that is "+

this.getSize()+". Special features to follow: \n"+this.getSpecial();

}

}

/// Car.java END ///

/// Toyota.java START ///

public class Toyota extends Car{

private String model, style, size, special;

public Toyota(String m, String st, String sz, String sp){

model = m;

style = st;

size = sz;

special = sp;

}

@Override

public String getModel(){

return this.model;

}

@Override

public String getStyle(){

return this.style;

}

@Override

public String getSize(){

return this.size;

}

@Override

public String getSpecial(){

return this.special;

}

public void slogan(){

System.out.println("Let's Go Places");

}

}

/// Toyota.java END ///

/// Ford.java START ///

public class Ford extends Car{

private String model, style, size, special;

public Ford(String m, String st, String sz, String sp){

model = m;

style = st;

size = sz;

special = sp;

}

@Override

public String getModel(){

return this.model;

}

@Override

public String getStyle(){

return this.style;

}

@Override

public String getSize(){

return this.size;

}

@Override

public String getSpecial(){

return this.special;

}

public void slogan(){

System.out.println("Built Ford Tough");

}

}

/// Ford.java END ///

/// Chevy.java START ///

public class Chevy extends Car{

private String model, style, size, special;

public Chevy(String m, String st, String sz, String sp){

model = m;

style = st;

size = sz;

special = sp;

}

@Override

public String getModel(){

return this.model;

}

@Override

public String getStyle(){

return this.style;

}

@Override

public String getSize(){

return this.size;

}

@Override

public String getSpecial(){

return this.special;

}

public void slogan(){

System.out.println("Together Let's Drive");

}

}

/// Chevy.java END ///

/// Nissan.java START ///

public class Nissan extends Car{

private String model, style, size, special;

public Nissan(String m, String st, String sz, String sp){

model = m;

style = st;

size = sz;

special = sp;

}

@Override

public String getModel(){

return this.model;

}

@Override

public String getStyle(){

return this.style;

}

@Override

public String getSize(){

return this.size;

}

@Override

public String getSpecial(){

return this.special;

}

public void slogan(){

System.out.println("Innovation That Excites");

}

}

/// Nissan.java END ///

/// CarFactory.java START ///

public class CarFactory {

public Car getCar(String makeName, String m, String st, String sz, String sp){

if (makeName.equalsIgnoreCase("toyota"))

return new Toyota(m, st, sz, sp);

else if (makeName.equalsIgnoreCase("Ford"))

return new Ford(m, st, sz, sp);

else if (makeName.equalsIgnoreCase("chevy"))

return new Chevy(m, st, sz, sp);

else if (makeName.equalsIgnoreCase("nissan"))

return new Nissan(m, st, sz, sp);

else

return null;

}

}

/// CarFactory.java END ///

Screenshot showing it working:

A screenshot of a computer

Description automatically generated