Lab Exercise: Collections

INTERTECH

Instructors Who Consult. | **Consultants** Who Teach.

About Intertech

Thank you for choosing Intertech for your training. The next page of this document will get you started with your lab if you'd like to skip ahead. Below is a brief overview of Intertech as well as a promo code for you for future live Intertech trainings.

Intertech (www.intertech.com) is the largest combined software developer **training** and **consulting** firm in Minnesota. Our unique blend of training, consulting, and mentoring has empowered technology teams in small businesses, Fortune 500 corporations and government agencies since 1991.

Our training organization offers live in-classroom and online deliveries, private on-site deliveries, and on-demand options. We cover a broad spectrum of .NET, Java, Agile/Scrum, Web Development, and Mobile technologies. See more information on our training and search for courses by **clicking here**.

We appreciate you choosing Intertech!

Lab Exercise

Collections

Collections provide an alternate to arrays. Collections are dynamic in length and serve as heterogeneous data structures. Methods are used to add, remove, and fetch data from these structures. Iterating through the elements of a collection also takes a bit more work.

In this lab, you explore some popular collection types to store a catalog of goods.

Specifically, in this lab you will:

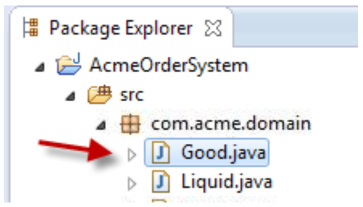
- Create and initialize an ArrayList
- Optionally create and initialize a HashSet
- Learn how to add, remove, and retrieve elements from a collection
- Explore how to iterate through a collection with Iterator
- Once again observe the power of polymorphism through a heterogeneous collection

Scenario

Given Acme's diverse set of product offerings, they need an electronic catalog of products. In this lab, you create a product catalog using a collection object.

Step 1: Add a catalog to the Good class

1.1 Add a new catalog static field to Good. In the Package Explorer view, double-click on the Good.java file in the com.acme.domain package to open the file in a Java editor.



Add a new field that contains an ArrayList of goods of either Solid or Liquid instances.

```
private static List catalog;
```

1.2 Create a getter method to encapsulate the catalog static field that retrieves and returns the collection of goods.

```
public static List getCatalog() {
  return catalog;
}
```

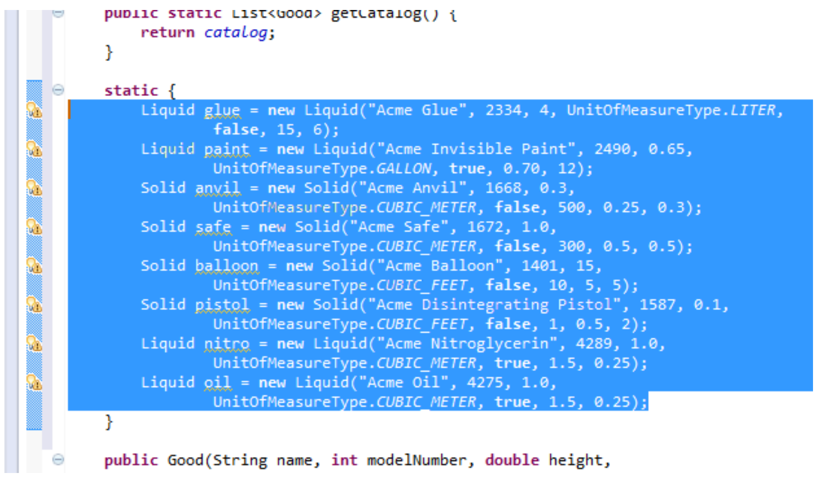
1.3 Create a static block to initialize the catalog.

1.3.1 First, create the static block in the Good.java file.

```
static {

}
```

1.3.2 Open the goods.txt file from the IntertechLearnJava zip (found in the Resources section for the “HelloWorld Lab”) with a text editor. This file contains the code to create several Solid and Liquid goods.

1.3.3 Copy and paste the contents of the goods.txt file into the static method.



```
    public static List<Good> getCatalog() {
        return catalog;
    }

    static {
        Liquid glue = new Liquid("Acme Glue", 2334, 4, UnitOfMeasureType.LITER,
                false, 15, 6);
        Liquid paint = new Liquid("Acme Invisible Paint", 2490, 0.65,
                UnitOfMeasureType.GALLON, true, 0.70, 12);
        Solid anvil = new Solid("Acme Anvil", 1668, 0.3,
                UnitOfMeasureType.CUBIC_METER, false, 500, 0.25, 0.3);
        Solid safe = new Solid("Acme Safe", 1672, 1.0,
                UnitOfMeasureType.CUBIC_METER, false, 300, 0.5, 0.5);
        Solid balloon = new Solid("Acme Balloon", 1401, 15,
                UnitOfMeasureType.CUBIC_FEET, false, 10, 5, 5);
        Solid pistol = new Solid("Acme Disintegrating Pistol", 1587, 0.1,
                UnitOfMeasureType.CUBIC_FEET, false, 1, 0.5, 2);
        Liquid nitro = new Liquid("Acme Nitroglycerin", 4289, 1.0,
                UnitOfMeasureType.CUBIC_METER, true, 1.5, 0.25);
        Liquid oil = new Liquid("Acme Oil", 4275, 1.0,
                UnitOfMeasureType.CUBIC_METER, true, 1.5, 0.25);
    }

    public Good(String name, int modelNumber, double height,
```

1.3.4 After the lines of code that create the eight goods, initialize the catalog with a new ArrayList.

```
catalog = new ArrayList();
```

1.3.5 Add each of the goods to the ArrayList. The example below adds “anvil” to the ArrayList of goods.

```
catalog.add(anvil);
```

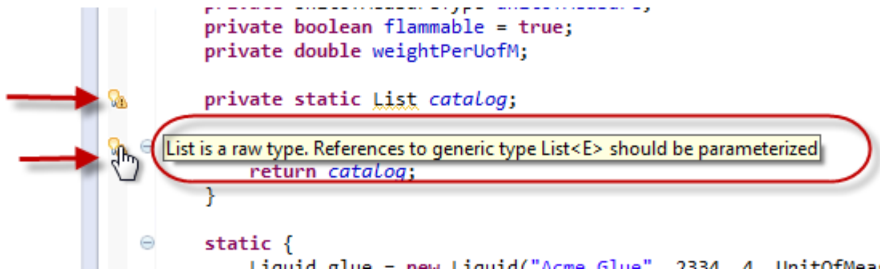
1.3.6 When complete, the static method should now look similar to the code below.

```
static {
  Liquid glue = new Liquid("Acme Glue", 2334, 4,
UnitOfMeasureType.LITER,  false, 15, 6);
  Liquid paint = new Liquid("Acme Invisible Paint", 2490, 0.65,
UnitOfMeasureType.GALLON, true, 0.70, 12);
  Solid anvil = new Solid("Acme Anvil", 1668, 0.3,
UnitOfMeasureType.CUBIC_METER, false, 500, 0.25, 0.3);
  Solid safe = new Solid("Acme Safe", 1672, 1.0,
UnitOfMeasureType.CUBIC_METER, false, 300, 0.5, 0.5);
  Solid balloon = new Solid("Acme Balloon", 1401, 15,
UnitOfMeasureType.CUBIC_FEET, false, 10, 5, 5);
  Solid pistol = new Solid("Acme Disintegrating Pistol", 1587,
0.1, UnitOfMeasureType.CUBIC_FEET, false, 1, 0.5, 2);
  Liquid nitro = new Liquid("Acme Nitroglycerin", 4289, 1.0,
UnitOfMeasureType.CUBIC_METER, true, 0.25, 1.5);
  Liquid oil = new Liquid("Acme Oil", 4275, 1.0,
UnitOfMeasureType.CUBIC_METER, true, 0.25, 1.5);
  catalog = new ArrayList();
  catalog.add(glue);
  catalog.add(paint);
  catalog.add(anvil);
  catalog.add(safe);
  catalog.add(balloon);
  catalog.add(pistol);
  catalog.add(nitro);
  catalog.add(oil);
}
```

1.4 Add import statements to Goods.java in order to use the needed java.util classes.

```
import java.util.ArrayList;
import java.util.List;
```

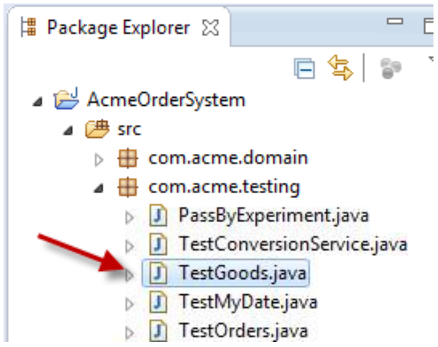
Note: After entering the code for the catalog, you may detect several compiler warnings (like that pictured below) in the Good.java file. This is ok. The issue about generics will be covered in the next chapter and the warnings will be removed with the next lab.



```
private boolean flammable = true;
private double weightPerUofM;

private static List catalog;

List is a raw type. References to generic type List<E> should be parameterized
    return catalog;
}

static {
```

Step 2: Test the catalog

2.1 Add code to TestGoods to test the catalog. In the Package Explorer view, double-click on the TestGoods.java file in the com.acme.testing package to open the file in a Java editor.



Add the line of code at the bottom of the main( ) method to call on and display the goods catalog.

```
System.out.println(Good.getCatalog());
```

2.2 Save your changes to Good and TestGoods classes. Make sure there are no compiler errors in either class. Run TestGoods and check the output.

```
...
[Acme Glue-2334 (liquid) 452.3893421169302 LITER, Acme Invisible
Paint-2490 (liquid) 294.05307237600465 GALLON, Acme Anvil-1668
that is 0.0225 CUBIC_METER in size, Acme Safe-1672 that is 0.25
CUBIC_METER in size, Acme Balloon-1401 that is 375.0 CUBIC_FEET
in size, Acme Disintegrating Pistol-1587 that is 0.1 CUBIC_FEET
in size, Acme Nitroglycerin-4289 (liquid) 7.0685834705770345
CUBIC_METER, Acme Oil-4275 (liquid) 7.0685834705770345
CUBIC_METER]
```

Note: You may note that passing the ArrayList object to the println( ) method of System.out results in a call to toString( ) on each of the ArrayList elements.

Step 3: Add/Remove Goods

In order to see the dynamic nature of a collection versus an array in Java, add and remove some goods to the catalog.

3.1 Remove the second good (Invisible Paint). Product engineers have determined a flaw in the Invisible Paint. Because this product doesn't really make you invisible, remove it from the catalog. At the bottom of the main( ) method in TestGoods.java, make a call to remove Invisible Paint.

```
Good.getCatalog().remove(1);
```

3.2 Add another good. At the bottom of the main( ) method in TestGoods.java, create another product, the Acme Toaster, and add it to the catalog. In fact, make a call to add it twice.

```
Solid toaster = new Solid("Acme Toaster", 1755, 0.75,
UnitOfMeasureType.CUBIC_FEET, false, 1.0, 1.0, 1.0);
Good.getCatalog().add(toaster);
Good.getCatalog().add(toaster);
```

3.3 Redisplay the catalog. After the removal and adding of goods to the catalog, add a line of code at the bottom of the main( ) method to display the goods catalog.

```
System.out.println(Good.getCatalog());
```

3.4 Save your changes to the TestGoods class. Make sure there are no compiler errors in the class. Run TestGoods and check the output. You should no longer see Acme Paint in the list of goods, but Toaster is displayed twice at the bottom of the list!

```
...
[Acme Glue-2334 (liquid) 452.3893421169302 LITER, Acme Anvil-
1668 that is 0.0225 CUBIC_METER in size, Acme Safe-1672 that is
0.25 CUBIC_METER in size, Acme Balloon-1401 that is 375.0
CUBIC_FEET in size, Acme Disintegrating Pistol-1587 that is 0.1
CUBIC_FEET in size, Acme Nitroglycerin-4289 (liquid)
7.0685834705770345 CUBIC_METER, Acme Oil-4275 (liquid)
7.0685834705770345 CUBIC_METER, Acme Toaster-1755 that is 0.75
CUBIC_FEET in size, Acme Toaster-1755 that is 0.75 CUBIC_FEET in
size]
```

Step 4: Iterate for flammables

For safety reasons, Acme often needs to access a list of flammable products quickly. Use a collection iterator to loop through the catalog and return a set of flammable goods.

4.1 Add a flammablesList( ) static method to Good.java. This new method iterates though the catalog collection and checks whether each product is flammable. If it is, put the flammable product in a new HashSet.

```
public static Set flammablesList() {
  Set flammables = new HashSet();
  Iterator i = Good.getCatalog().iterator();
  while (i.hasNext()) {
    Good x = (Good) i.next();
    if (x.isFlammable()) {
      flammables.add(x);
    }
  }
  return flammables;
}
```

4.2 Import the necessary classes. The last method requires you add the following imports to Good.java.

```
import java.util.Iterator;
import java.util.HashSet;
import java.util.Set;
```

4.3 At the bottom of the main( ) method in TestGoods.java, display the flammable list.

```
System.out.println("Flammable products:  " +
Good.flammablesList());
```

4.4 Save your changes to the Good and TestGoods classes. Make sure there are no compiler errors in either class. Run TestGoods and check to see that the appropriate flammables are listed.

```
...
Flammable products:  [Acme Nitroglycerin-4289 (liquid)
7.0685834705770345 CUBIC_METER, Acme Oil-4275 (liquid)
7.0685834705770345 CUBIC_METER]
```

Lab Solutions

Good.java

```
package com.acme.domain;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

public abstract class Good implements Product {
  public enum UnitOfMeasureType {
    LITER, GALLON, CUBIC_METER, CUBIC_FEET
  }

  private String name;
  private int modelNumber;
  private double height;
  private UnitOfMeasureType unitOfMeasure;
  private boolean flammable = true;
  private double weightPerUofM;

  private static List catalog;

  public static List getCatalog() {
    return catalog;
  }

  static {
    Liquid glue = new Liquid("Acme Glue", 2334, 4, 
UnitOfMeasureType.LITER,
        false, 15, 6);
    Liquid paint = new Liquid("Acme Invisible Paint", 2490, 
0.65,
        UnitOfMeasureType.GALLON, true, 0.70, 12);
    Solid anvil = new Solid("Acme Anvil", 1668, 0.3,
        UnitOfMeasureType.CUBIC_METER, false, 500, 0.25, 0.3);
    Solid safe = new Solid("Acme Safe", 1672, 1.0,
        UnitOfMeasureType.CUBIC_METER, false, 300, 0.5, 0.5);
    Solid balloon = new Solid("Acme Balloon", 1401, 15,
        UnitOfMeasureType.CUBIC_FEET, false, 10, 5, 5);
    Solid pistol = new Solid("Acme Disintegrating Pistol", 1587, 
0.1,
        UnitOfMeasureType.CUBIC_FEET, false, 1, 0.5, 2);
    Liquid nitro = new Liquid("Acme Nitroglycerin", 4289, 1.0,
        UnitOfMeasureType.CUBIC_METER, true, 0.25, 1.5);
    Liquid oil = new Liquid("Acme Oil", 4275, 1.0,
        UnitOfMeasureType.CUBIC_METER, true, 0.25, 1.5);
    catalog = new ArrayList();
    catalog.add(glue);
    catalog.add(paint);
    catalog.add(anvil);
    catalog.add(safe);
    catalog.add(balloon);
    catalog.add(pistol);
    catalog.add(nitro);
    catalog.add(oil);
  }

  public static Set flammablesList() {
    Set flammables = new HashSet();
    Iterator i = Good.getCatalog().iterator();
    while (i.hasNext()) {
      Good x = (Good) i.next();
      if (x.isFlammable()) {
        flammables.add(x);
      }
    }
    return flammables;
  }

  public Good(String name, int modelNumber, double height,
      UnitOfMeasureType uoM, boolean flammable, double
wgtPerUoM) {
    this.name = name;
    this.modelNumber = modelNumber;
    this.height = height;
    this.unitOfMeasure = uoM;
    this.flammable = flammable;
    this.weightPerUofM = wgtPerUoM;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public int getModelNumber() {
    return modelNumber;
  }

  public void setModelNumber(int modelNumber) {
    this.modelNumber = modelNumber;
  }

  public double getHeight() {
    return height;
  }

  public void setHeight(double height) {
    this.height = height;
  }

  public UnitOfMeasureType getUnitOfMeasure() {
    return unitOfMeasure;
  }

  public void setUnitOfMeasure(UnitOfMeasureType unitOfMeasure)
{
    this.unitOfMeasure = unitOfMeasure;
  }

  public boolean isFlammable() {
    return flammable;
  }

  public void setFlammable(boolean flammable) {
    this.flammable = flammable;
  }

  public double getWeightPerUofM() {
    return weightPerUofM;
  }

  public void setWeightPerUofM(double weightPerUofM) {
    this.weightPerUofM = weightPerUofM;
  }

  public String toString() {
    return name + "-" + modelNumber;
  }

  public abstract double volume();

  public double weight() {
    return volume() * weightPerUofM;
  }
}
```

TestGoods.java

```
package com.acme.testing;

import com.acme.domain.Good;
import com.acme.domain.Good.UnitOfMeasureType;
import com.acme.domain.Liquid;
import com.acme.domain.Solid;

public class TestGoods {

  public static void main(String[] args) {
    Liquid glue = new Liquid("Acme Glue", 2334, 4,
UnitOfMeasureType.LITER,
        false, 15, 6);
    Liquid paint = new Liquid("Acme Invisible Paint", 2490,
0.65,
        UnitOfMeasureType.GALLON, true, 0.70, 12);
    Solid anvil = new Solid("Acme Anvil", 1668, 0.3,
        UnitOfMeasureType.CUBIC_METER, false, 5000, 0.5, 0.5);

    System.out.println(glue);
    System.out.println(paint);
    System.out.println(anvil);

    System.out.println("The weight of " + glue + " is " +
      glue.weight());
    System.out.println("The weight of " + paint + " is " +
      paint.weight());
    System.out.println("The weight of " + anvil + " is " +
      anvil.weight());

    Good x = glue;
    System.out.println("Is " + x + " flammable?  " +
x.isFlammable());
    x = paint;
    System.out.println("Is " + x + " flammable?  " +
x.isFlammable());

    System.out.println(Good.getCatalog());
    Good.getCatalog().remove(1);
    Solid toaster = new Solid("Acme Toaster", 1755, 0.75,
        UnitOfMeasureType.CUBIC_FEET, false, 1.0, 1.0, 1.0);
    Good.getCatalog().add(toaster);
    Good.getCatalog().add(toaster);
    System.out.println(Good.getCatalog());

    System.out.println("Flammable products:  " +
      Good.flammablesList());
    }
}
```

Bonus Lab

Step 5: Use HashSet instead of ArrayList

The benefit of using many of the Collection classes is that they are largely interchangeable. Use a HashSet instead of an ArrayList in the Good.java code.

What code has to change? What code stays the same? How do you remove paint? What happens when two toasters are added to the catalog?

Give Intertech a call at **1.800.866.9884** or visit Intertech's website.