Lab Exercise: Arrays

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

Arrays

Arrays provide a fixed-length data structure to collect any type of object or primitive in Java. Iterating through the elements of an array is relatively simple. Given its fixed length, a simple loop can be used to iterate through each element of the indexed array.

In this lab, you will explore an array to store MyDate objects.

Specifically, in this lab you will:

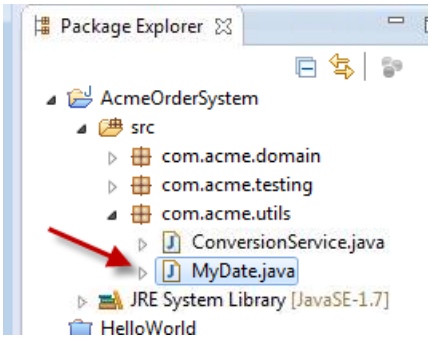
- Create and initialize an array
- Learn how to add elements to an array
- Explore how to iterate through an array with a for loop

Scenario

Acme, like all companies, takes holidays off. Therefore, orders created in the Acme Order System should not contain a MyDate that represents a holiday. You need to modify the Acme Order System to create an array of holiday MyDates and check each order to ensure that is does not contain an order date on one of these dates.

Step 1: Add an array of holidays to MyDate

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



Add a new field that contains an array of MyDate instances that represent the six holidays of the year.

```
private static MyDate[] holidays;
```

1.2 Create a static block to initialize the holidays array. There are six holidays in the Acme calendar.

New Year's Date (January 1)	Labor Day (Sept 5)
Memorial Day (May 30)	Thanksgiving (November 24)

Independence Day (July 4)	Christmas (December 25)

Note: these are the holiday dates for 2016. Feel free to modify to add holidays for any year of your choice. If you do add different dates, note that your output will be different.

In the static block, initialize the array for six elements, and set each of the array's elements to a MyDate for each holiday listed above. Part of the static block is completed below.

```
static {
  holidays = new MyDate[6];
  //... complete the initialization of the holidays array.
}
```

1.3 Create a getter method to encapsulate the holidays' static fields and retrieve and return the MyDate holidays array.

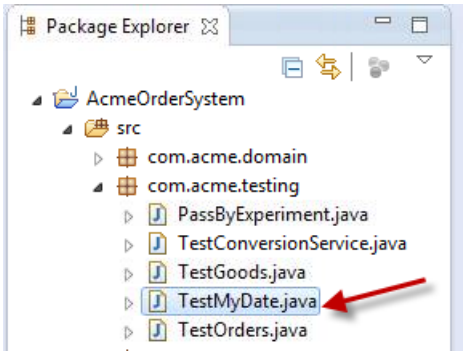
```
public static MyDate[] getHolidays(){
  return holidays;
}
```

1.4 Create a method to loop through and display the company holidays.

```
public static void listHolidays() {
  System.out.println("the holidays are:");
  for (int x = 0; x < holidays.length; x++){
    System.out.println(holidays[x]);
  }
}
```

Step 2: Test the holiday array

2.1 Add a call to listHolidays( ) in TestMyDate. Open TestMyDates.java.



Add a line of code to the end of the main method to list the holidays.

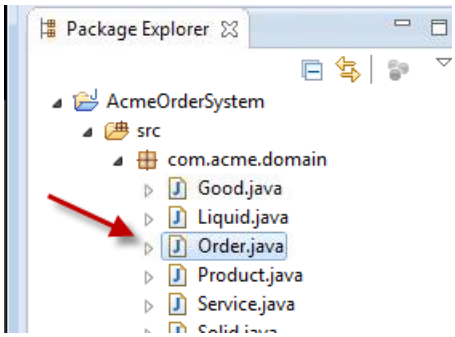
```
MyDate.listHolidays();
```

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

```
Attempting to create a non-valid date 13/40/-1
11/11/1918
11/11/1918
1/1/1900
The year 1752 is a leap year
...
The year 2020 is a leap year
These two dates are equal
the holidays are:
1/1/2012
5/28/2012
7/4/2012
9/3/2012
11/22/2012
12/25/2012
```

Step 3: Modify Order

3.1 Modify the Order class to disallow holiday order dates. Open the Order class found in the com.acme.domain.



Add a method to test whether the proposed order date is a holiday or not.

```
private boolean isHoliday(MyDate proposedDate) {
  for (int x = 0; x < MyDate.getHolidays().length; x++) {
    MyDate holiday = (MyDate) MyDate.getHolidays()[x];
    if ((holiday.getDay() == proposedDate.getDay())
        && (holiday.getMonth() == proposedDate.getMonth())
        && (holiday.getYear() == proposedDate.getYear())) {
      return true;
    }
  }
  return false;
}
```

3.2 Modify the setOrderDate( ) method on Order to check for and disallow proposed order dates that fall on a holiday.

```
public void setOrderDate(MyDate orderDate) {
  if (isHoliday(orderDate)) {
    System.out.println("Order date, " + orderDate + ", cannot be
set to a holiday!");
  } else {
    this.orderDate = orderDate;
  }
}
```

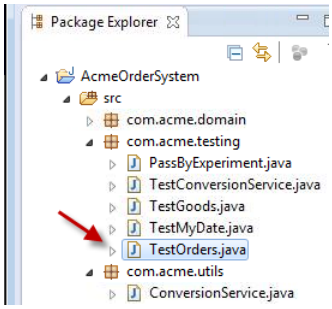
3.3 Modify the Order constructor to use the setOrderDate( ) method to set the order's date field.

```
public Order(MyDate d, double amt, String c, Product p, int q) {
  setOrderDate(d);
```

```
    orderAmount = amt;
    customer = c;
    product = p;
    quantity = q;
}
```

Step 4: Test Orders

4.1 Use TestOrders to check that orders cannot be created with a holiday date. Find and open TestOrders.java from the com.acme.testing package.



Modify one of the MyDate objects created in the main( ) method of TestOrders to fall on a holiday.

```
MyDate date3 = new MyDate(1,1,2016);
```

4.2 Save your changes to Order and TestOrders, and make sure there are no compiler errors. Then run TestOrders and check the output.

```
Attempting to set the quantity to a value less than or equal to
zero
10 ea. Acme Anvil-1668 that is 0.0225 CUBIC_METER in size for
Wile E Coyote
125 ea. Acme Balloon-1401 that is 375.0 CUBIC_FEET in size for
Bugs Bunny
The tax Rate is currently: 0.05
The tax for 3000.0 is: 150.0
The tax for this order is: 100.0
The tax for this order is: 50.0
The tax Rate is currently: 0.06
The tax for 3000.0 is: 180.0
The tax for this order is: 120.0
The tax for this order is: 60.0
The total bill for: 10 ea. Acme Anvil-1668 that is 0.0225
CUBIC_METER in size for Wile E Coyote is 2000.0
The tax for this order is: 60.0
The total bill for: 125 ea. Acme Balloon-1401 that is 375.0
CUBIC_FEET in size for Bugs Bunny is 1040.0
Order date, 1/1/2016, cannot be set to a holiday!
The total bill for: 1 ea. Road Runner Eradication(a 14 day
service) for Daffy Duck is 20000.0
```

Lab Solutions

MyDate.java

```
package com.acme.utils;

public class MyDate {
  // Member/instance variables (a.k.a.
fields/properties/attributes)
  private byte day;
  private byte month;
  private short year;

  private static MyDate[] holidays;

  static {
    holidays = new MyDate[6];
    holidays[0] = new MyDate(1, 1, 2016);
    holidays[1] = new MyDate(5, 30, 2016);
    holidays[2] = new MyDate(7, 4, 2016);
    holidays[3] = new MyDate(9, 5, 2016);
    holidays[4] = new MyDate(11, 24, 2016);
    holidays[5] = new MyDate(12, 25, 2016);
  }

  // Constructors:
  // 1. Same name as the class
  // 2. No return type

  // The no-args constructor
  public MyDate() {
    this(1, 1, 1900);
  }

  // Constructor that takes 3 arguments
  public MyDate(int m, int d, int y) {
    setDate(m, d, y);
  }

  // Methods
  public String toString() {
    return month + "/" + day + "/" + year;
  }

  public void setDate(int m, int d, int y) {
    if (valid(d, m, y)) {
      day = (byte) d;
      year = (short) y;
```

```
      month = (byte) m;
    }
  }

  public static void leapYears() {
    for (int i = 1752; i <= 2020; i = i + 4) {
      if (((i % 4 == 0) && (i % 100 != 0)) || (i % 400 == 0))
        System.out.println("The year " + i + " is a leap year");
    }
  }

  public int getDay() {
    return day;
  }

  public void setDay(int day) {
    if (valid(day, month, year)) {
      this.day = (byte) day;
    }
  }

  public int getMonth() {
    return month;
  }

  public void setMonth(int month) {
    if (valid(day, month, year)) {
      this.month = (byte) month;
    }
  }

  public int getYear() {
    return year;
  }

  public void setYear(int year) {
    if (valid(day, month, year)) {
      this.year = (short) year;
    }
  }

  private boolean valid(int day, int month, int year) {
    if (day > 31 || day < 1 || month > 12 || month < 1) {
      System.out.println("Attempting to create a non-valid date
" +
        month + "/" + day + "/" + year);
      return false;
    }
    switch (month) {
    case 4:
```

```
    case 6:
    case 9:
    case 11:
      return (day <= 30);
    case 2:
      return day <= 28 || (day == 29 && year % 4 == 0);
    }
    return true;
  }

  public boolean equals(Object o) {
    if (o instanceof MyDate) {
      MyDate d = (MyDate) o;
      if ((d.day == day) && (d.month == month) && (d.year ==
year)) {
        return true;
      }
    }
    return false;
  }
  public static MyDate[] getHolidays(){
    return holidays;
  }

  public static void listHolidays() {
    System.out.println("the holidays are:");
    for (int x = 0; x < holidays.length; x++){
      System.out.println(holidays[x]);
    }
  }
}
```

TestMyDate.java

```
package com.acme.testing;

import com.acme.utils.MyDate;

public class TestMyDate {

  public static void main(String[] args) {
    MyDate date1 = new MyDate(11, 11, 1918);

    MyDate date2 = new MyDate();
    date2.setDay(11);
    date2.setMonth(11);
    date2.setYear(1918);

    MyDate date3 = new MyDate();
    date3.setDate(13, 40, -1);

    String str1 = date1.toString();
    String str2 = date2.toString();
    String str3 = date3.toString();

    System.out.println(str1);
    System.out.println(str2);
    System.out.println(str3);

    MyDate.leapYears();

    MyDate newYear = new MyDate(1, 1, 2016);
    MyDate fiscalStart = new MyDate(1, 1, 2016);
    if (newYear.equals(fiscalStart))
      System.out.println("These two dates are equal");
    else
      System.out.println("These two dates are not equal");
    MyDate endOfYear = new MyDate(12,31,2016);
    if (newYear.equals(endOfYear))
      System.out.println("These two dates are equal");
    else
      System.out.println("These two dates are not equal");

    MyDate.listHolidays();

  }
}
```

Order.java

```
package com.acme.domain;

import com.acme.utils.MyDate;

public class Order {
  private MyDate orderDate;
  private double orderAmount = 0.00;
  private String customer;
  private Product product;
  private int quantity;

  public MyDate getOrderDate() {
    return orderDate;
  }

  public void setOrderDate(MyDate orderDate) {
    if (isHoliday(orderDate)) {
      System.out.println("Order date, " + orderDate + ", cannot
be set
        to a holiday!");
    } else {
      this.orderDate = orderDate;
    }
  }

  public double getOrderAmount() {
    return orderAmount;
  }

  public void setOrderAmount(double orderAmount) {
    if (orderAmount > 0) {
      this.orderAmount = orderAmount;
    } else {
      System.out
          .println("Attempting to set the orderAmount to a value
less
             than or equal to zero");
    }
  }

  public String getCustomer() {
    return customer;
  }

  public void setCustomer(String customer) {
    this.customer = customer;
  }
```

```
  public Product getProduct() {
    return product;
  }

  public void setProduct(Product product) {
    this.product = product;
  }

  public int getQuantity() {
    return quantity;
  }

  public void setQuantity(int quantity) {
    if (quantity > 0) {
      this.quantity = quantity;
    } else {
      System.out
          .println("Attempting to set the quantity to a value
less
             than or equal to zero");
    }
  }

  public static double getTaxRate() {
    return taxRate;
  }

  public static double taxRate = 0.05;

  public static void setTaxRate(double newRate) {
    taxRate = newRate;
  }

  public static void computeTaxOn(double anAmount) {
    System.out.println("The tax for " + anAmount + " is: " +
anAmount
        * Order.taxRate);
  }

  public Order(MyDate d, double amt, String c, Product p, int q)
{
    setOrderDate(d);
    orderAmount = amt;
    customer = c;
    product = p;
    quantity = q;
  }

  public String toString() {
```

```
    return quantity + " ea. " + product + " for " + customer;
  }

  public double computeTax() {
    System.out.println("The tax for this order is: " +
orderAmount
        * Order.taxRate);
    return orderAmount * Order.taxRate;
  }

  public char jobSize() {
    if (quantity <= 25) {
      return 'S';
    } else if (quantity <= 75) {
      return 'M';
    } else if (quantity <= 150) {
      return 'L';
    }
    return 'X';
  }

  public double computeTotal() {
    double total = orderAmount;
    switch (jobSize()) {
    case 'M':
      total = total - (orderAmount * 0.01);
      break;
    case 'L':
      total = total - (orderAmount * 0.02);
      break;
    case 'X':
      total = total - (orderAmount * 0.03);
      break;
    }
    if (orderAmount <= 1500) {
      total = total + computeTax();
    }
    return total;
  }

  private boolean isHoliday(MyDate proposedDate) {
    for (int x = 0; x < MyDate.getHolidays().length; x++) {
      MyDate holiday = (MyDate) MyDate.getHolidays()[x];
      if ((holiday.getDay() == proposedDate.getDay())
          && (holiday.getMonth() == proposedDate.getMonth())
          && (holiday.getYear() == proposedDate.getYear())) {
        return true;
      }
    }
    return false;
```

```
    }
}
```

TestOrders.java

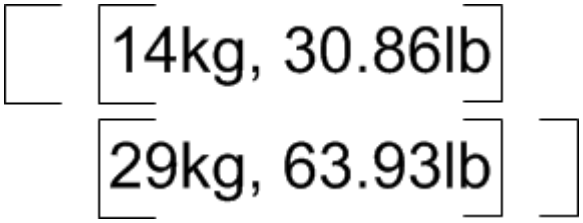
```
package com.acme.testing;

import com.acme.domain.Order;
import com.acme.domain.Service;
import com.acme.domain.Solid;
import com.acme.domain.Good.UnitOfMeasureType;
import com.acme.utils.MyDate;

public class TestOrders {

  public static void main(String[] args) {
    MyDate date1 = new MyDate(1, 20, 2008);
    Solid s1 = new Solid("Acme Anvil", 1668, 0.3,
      UnitOfMeasureType.CUBIC_METER, false, 500, 0.25, 0.3);
    Order anvil = new Order(date1, 2000.00, "Wile E Coyote", s1,
10);

    MyDate date2 = new MyDate(4, 10, 2008);
    Solid s2 = new Solid("Acme Balloon", 1401, 15,
      UnitOfMeasureType.CUBIC_FEET, false, 10, 5, 5);
    Order balloons = new Order(date2, 1000.00, "Bugs Bunny", s2,
125);
    balloons.setQuantity(-200);

    System.out.println(anvil);
    System.out.println(balloons);

    System.out.println("The tax Rate is currently: " +
Order.taxRate);
    Order.computeTaxOn(3000.00);
    anvil.computeTax();
    balloons.computeTax();

    Order.setTaxRate(0.06);
    System.out.println("The tax Rate is currently: " +
Order.taxRate);
    Order.computeTaxOn(3000.00);
    anvil.computeTax();
    balloons.computeTax();
    System.out.println("The total bill for: " + anvil + " is "
        + anvil.computeTotal());
    System.out.println("The total bill for: " + balloons + " is
"
        + balloons.computeTotal());

    MyDate date3 = new MyDate(1,1,2012);
```

```
    Service s3 = new Service("Road Runner Eradication", 14,
false);
    Order birdEradication = new Order(date3, 20000, "Daffy
Duck", s3,
      1);
    System.out.println("The total bill for: " + birdEradication
+ " is
      " + birdEradication.computeTotal());
  }
}
```

Bonus Lab

4.3 Modify the listHolidays( ) method of the MyDate class to use a for-each loop. This should not affect the output or display of the method.

4.4 Modify the ConversionService class. Add a static method that converts any number (variable argument) of kilogram measures to pounds. The method should return an array containing an array of the original kilogram value conversion pound value. Say, for example, the method was passed 14 and 29 kilograms, then the method would return a multidimensional array containing the pairs of kg to lb conversion values as shown below.



4.4.1 Here is the start of the conversion method.

```
public static double[][] allKgToPounds(double... kilogramValues)
{
   //  … code here
}
```

4.4.2 Add lines of code to the end of the main( ) method in TestConversionService to check how well your new conversion service works. The method should pass several kilogram values (as doubles) to the method. Capture the results in a multi-dimensional array and print out the results.

```
double[][] results = ConversionService.allKgToPounds(14.0, 29.0,
6.5, 7.7);
for (double[] result : results) {
  System.out.println(result[0] + "->" + result[1]);
}
```

4.4.3 What happens if a set of integers are passed to the method? Add another set of lines to the main( ) method of TestConversionService to pass a variable set of ints to the same method. Does it work? If so, do you know why it still works?

```
results = ConversionService.allKgToPounds(2, 5, 8, 4);
for (double[] result : results) {
  System.out.println(result[0] + "->" + result[1]);
}
```

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