 JMU - Department of Computer Science

[Help](https://w3.cs.jmu.edu/bernstdh/Web/common/help/) [Tools](https://w3.cs.jmu.edu/bernstdh/Web/common/tools/)

Lab: Experimenting with Interfaces

**Instructions:** Answer the following questions one at a time. After answering each question, check your answer (by clicking on the check-mark icon if it is available) before proceeding to the next question.

**Getting Ready:** Before going any further, you should:

1. Depending on your development environment, [create](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/#) either a directory or a project for this lab.
2. [Setup](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/#) your development environment.
3. [Download](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/#) the following files:  
     
   [HolidayAccount.java](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/HolidayAccount.java)  
   [Driver1.java](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/Driver1.java)to an appropriate directory/folder. (In most browsers/OSs, the easiest way to do this is by right-clicking/control-clicking on each of the links above.)

*1. Getting Started with Interfaces:*  This part of the lab will get you started with interfaces.

1. Read the source code for the HolidayAccount class and Driver1.
2. Compile and execute Driver1.
3. What output was generated?  
   .  
   HA-0000000003 user0  
   HA-0000000002 user1  
   HA-0000000001 user2  
   HA-0000000000 user3  
   
4. Add the following to the end of the main() method in Driver1:  
    // Sort the accounts  
    Arrays.sort(accounts);  
     
     
    // Print the accounts in the sorted order  
    for (i=0; i < accounts.length; i++)  
    {  
    System.out.println(accounts[i].getAccountID()+"\t"+  
    accounts[i].getHolder());  
    }
5. Compile and execute Driver1.
6. What [error](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/#) message was generated?  
   .  
   Exception in thread "main" java.lang.ClassCastException: HolidayAccount  
    at java.util.Arrays.mergeSort(Unknown Source)  
    at java.util.Arrays.sort(Unknown Source)  
    at Driver1.main(Driver1.java:44)  
   
7. Read the documentation for the Arrays.sort(java.lang.Object[])  method.
8. What interface must all elements in the Object array implement?  
   .  
   The Comparable interface.
9. Why was this error message generated?  
   .  
   The error message above was generated because the Arrays.sort(java.lang.Object[])  method (and other methods it calls) expects to be passed objects that implement the Comparable  interface. The HolidayAccount class doesn't.
10. Does the HolidayAccount class contain a compareTo() method?  
    .  
    Yes.
11. Given the answer to the previous questions, it seems like this defect can be corrected by declaring that the HolidayAccount class realizes the Comparable interface. What change do you need to make to accomplish this?  
    .  
    The declaration of the class needs to be changed to:  
    public class HolidayAccount implements Comparable  
    
12. Make this change and compile HolidayAccount.
13. What error message was generated?  
    .  
    HolidayAccount.java:10: HolidayAccount is not abstract and does not override   
    abstract method compareTo(java.lang.Object) in java.lang.Comparable  
    public class HolidayAccount implements Comparable  
     ^  
    
14. Why was this error message generated? In other words, why does the HolidayAccount class not actually implement the Comparable interface?  
    .  
    This error message was generated because the Comparable interface requires classes to have a public int compareTo(Object otherObject) method and the HolidayAccount class actually has a public int compareTo(HolidayAccount other) method (i.e., the parameters are of different type).
15. Fix this problem by changing the type of the parameter of the compareTo() method and casting it to a HolidayAccount within the method.
16. What changes did you make?  
    .  
     public int compareTo(Object otherObject)  
     {  
     HolidayAccount other;  
     int relation;  
      
     other = (HolidayAccount)otherObject;  
      
    
17. Compile and execute Driver1.
18. What output was generated?  
    .  
    HA-0000000003 user0  
    HA-0000000002 user1  
    HA-0000000001 user2  
    HA-0000000000 user3  
      
      
      
    HA-0000000000 user3  
    HA-0000000001 user2  
    HA-0000000002 user1  
    HA-0000000003 user0  
    

*2. Getting More Sophisticated:*  This part of the lab will help you think about interfaces in a more sophisticated way and introduce the importance of a topic that you will study later.

1. "Undo" the changes you just made to the compareTo(HolidayAccount other) method. In other words, change it back to:  
    public int compareTo(HolidayAccount other)  
    {  
    int relation;  
     
    relation = 0;  
    if (this.accountNumber < other.accountNumber)  
    {  
    relation = -1;  
    }   
    else if (this.accountNumber > other.accountNumber)  
    {  
    relation = 1;  
    }  
     
    return relation;  
    }
2. Add the following method to the HolidayAccount class:  
    /\*\*  
    \* Compare the account number on this account to the  
    \* account number on a given account (required by Comparable)  
    \*  
    \* @param other The given account  
    \* @return -1/1 if this [account](https://users.cs.jmu.edu/bernstdh/Web/common/labs/experimenting_interfaces/#) comes before/after the given account  
    \*/  
    public int compareTo(Object other)  
    {  
    return compareTo(other);   
    }
3. Does this class now implement the Comparable interface?  
   .  
   Yes.
4. Compile HolidayAccount.
5. Does HolidayAccount contain any syntax errors?  
   .  
   No.
6. Compile Driver1.
7. Does Driver1 contain any syntax errors?  
   .  
   No.
8. Execute Driver1.
9. What run-time error message was generated?  
   .  
   Exception in thread "main" java.lang.StackOverflowError  
   
10. Why was this error message generated?  
    .  
    This error message was generated because the compareTo(Object) method calls itself, creating an infinite recursion.
11. Fix this problem by typecasting other in such a way that the compareTo(Object) method calls the appropriate implementation.
12. What change(s) did you make?  
    .  
     return compareTo((HolidayAccount)other);   
    
13. Compile HolidayAccount. Compile and execute Driver1.
14. What output was generated?  
    .  
    HA-0000000003 user0  
    HA-0000000002 user1  
    HA-0000000001 user2  
    HA-0000000000 user3  
      
      
      
    HA-0000000000 user3  
    HA-0000000001 user2  
    HA-0000000002 user1  
    HA-0000000003 user0  
    
15. Which *working* implementation of the HolidayAccount class do you prefer, the one with just the compareTo(Object) method, or the one with both a compareTo(HolidayAccount) method and compareTo(Object) method? Why?  
    .  
    You might think that the implementation with two methods is preferred because it doesn't "break" any other classes that might have used the original HolidayAccount class. However, it's important to realize that any other classes that called the compareTo(HolidayAccount) method will continue to work even if there is just a compareTo(Object) method because a HolidayAccount "is an" Object.  
    I prefer the implementation with two methods because it contains a type safe compareTo() method that other classes can (and should be encouraged to) use.  
    We will see how this defect can be corrected later in the semester.

*3. Implementing Other Kinds of Contracts:*  This part of the lab will help you realize that some contracts do not involve interfaces.

1. Modify the HolidayAccount class so that it has an equals() method that is passed a Object object named other. Your implementation must call compareTo() and must not use an if statement or loop.
2. What code did you add?  
   .  
    /\*\*  
    \* Indicates whether some other object "is equal to" this one.  
    \*  
    \* @param other The other Object  
    \* @return true if the two are equal  
    \*/  
    public boolean equals(Object other)  
    {  
    return (compareTo((HolidayAccount)other) == 0);  
    }  
   
3. Read the requirements of the equals() method in the Object  class. Does your implementation satisfy all of these requirements?  
   .  
   No, it does not satsify the requirement for when other is null. In fact, if other is null, this implementation will throw a NullPointerException.
4. Modify your equals() method accordingly. How did you implement it now?  
   .  
    /\*\*  
    \* Indicates whether some other object "is equal to" this one.  
    \*  
    \* @param other The other Object  
    \* @return true if the two are equal  
    \*/  
    public boolean equals(Object other)  
    {  
    if (other == null) return false;  
    else return (compareTo((HolidayAccount)other) == 0);  
    }  
   

[Department of Computer Science](http://www.cs.jmu.edu/)

Copyright 2019