Relational Databases with MySQL Week 11 Assignment

Points possible: 70

Livermore, Tyler

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: Complete the coding steps. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push the Java project to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

1. Create a class of whatever type you want (Animal, Person, Camera, Cheese, etc.).
    a. Do not implement the Comparable interface.
    b. Add a name instance variable so that you can tell the objects apart.
    c. Add getters, setters and/or a constructor as appropriate.
    d. Add a toString method that returns the name and object type (like "Pentax Camera").
    e. Create a static method named `compare` in the class that returns an int and takes two of the objects as parameters. Return -1 if parameter 1 is "less than" parameter 2. Return 1 if parameter 1 is "greater than" parameter 2. Return 0 if the two parameters are "equal".
    f. Create a static list of these objects, adding at least 4 objects to the list.
    g. In another class, write a method to sort the objects using a Lambda expression using the compare method you created earlier.
    h. Write a method to sort the objects using a Method Reference to the compare method you created earlier.

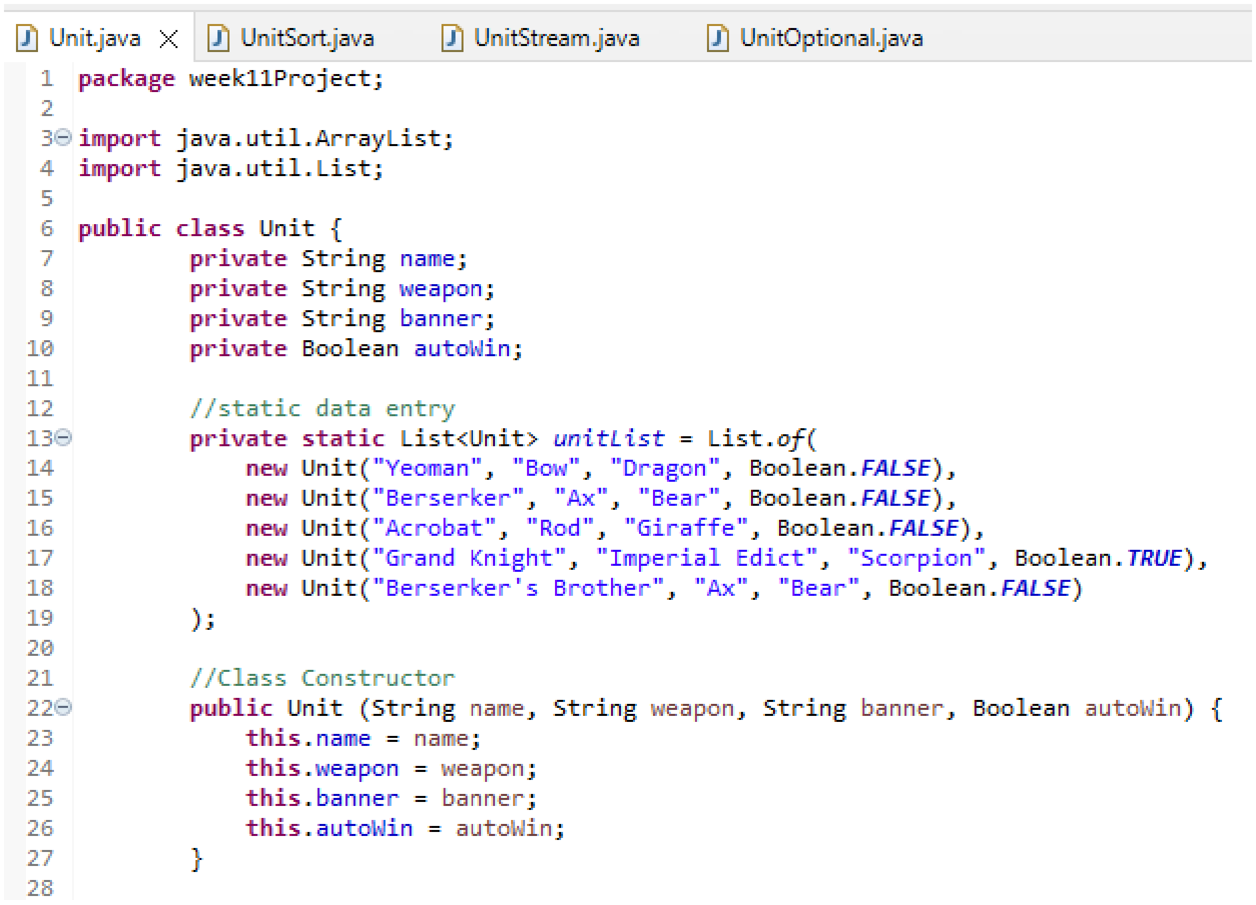
    i. Create a main method to call the sort methods.
    j. Print the list after sorting (System.out.println).

2. Create a new class with a main method. Using the list of objects you created in the prior step.
    a. Create a Stream from the list of objects.
    b. Turn the Stream of object to a Stream of String (use the map method for this).
    c. Sort the Stream in the natural order. (Note: The String class implements the Comparable interface, so you won't have to supply a Comparator to do the sorting.)
    d. Collect the Stream and return a comma-separated list of names as a single String. Hint: use Collectors.joining(", ") for this.
    e. Print the resulting String.
3. Create a new class with a main method. Create a method (method a) that accepts an Optional of some type of object (Animal, Person, Camera, etc.).
    a. The method should return the object unwrapped from the Optional if the object is present. For example, if you have an object of type Cheese, your method signature should look something like this:

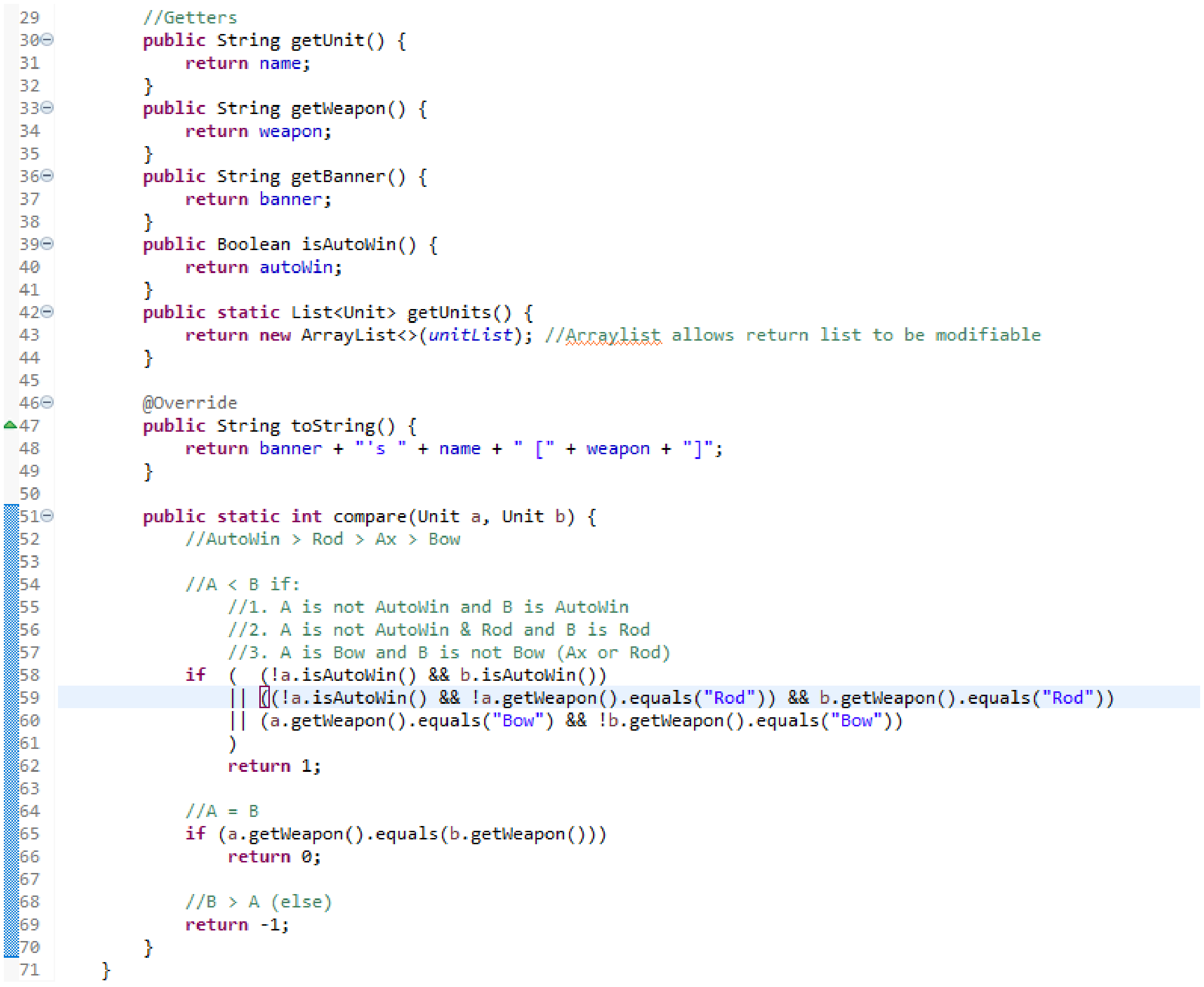
       ```
       public Cheese cheesyMethod(Optional<Cheese> optionalCheese) {...}
       ```

    b. The method should throw a NoSuchElementException with a custom message if the object is not present.
    c. Create another method (method b) that calls method a with an object wrapped by an Optional. Show that the object is returned unwrapped from the Optional (i.e., print the object).
    d. Method b should also call method a with an empty Optional. Show that a NoSuchElementException is thrown by method a by printing the exception message. Hint: catch the `NoSuchElementException` as parameter named "e" and do `System.out.println(e.getMessage())`.
    e. Note: your method should handle the Optional as shown in the video on Optionals using the orElseThrow method. For the missing object, you must use a Lambda expression in orElseThrow to return a NoSuchElementException with a custom message.

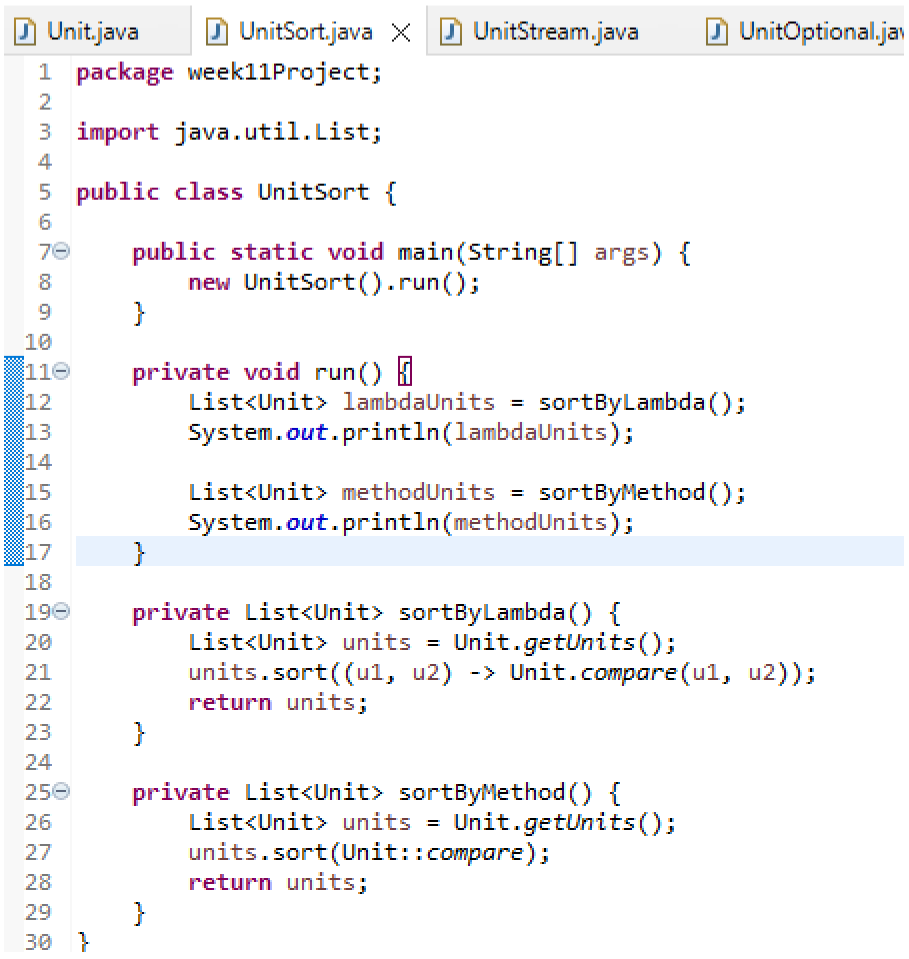
Screenshots of Code:



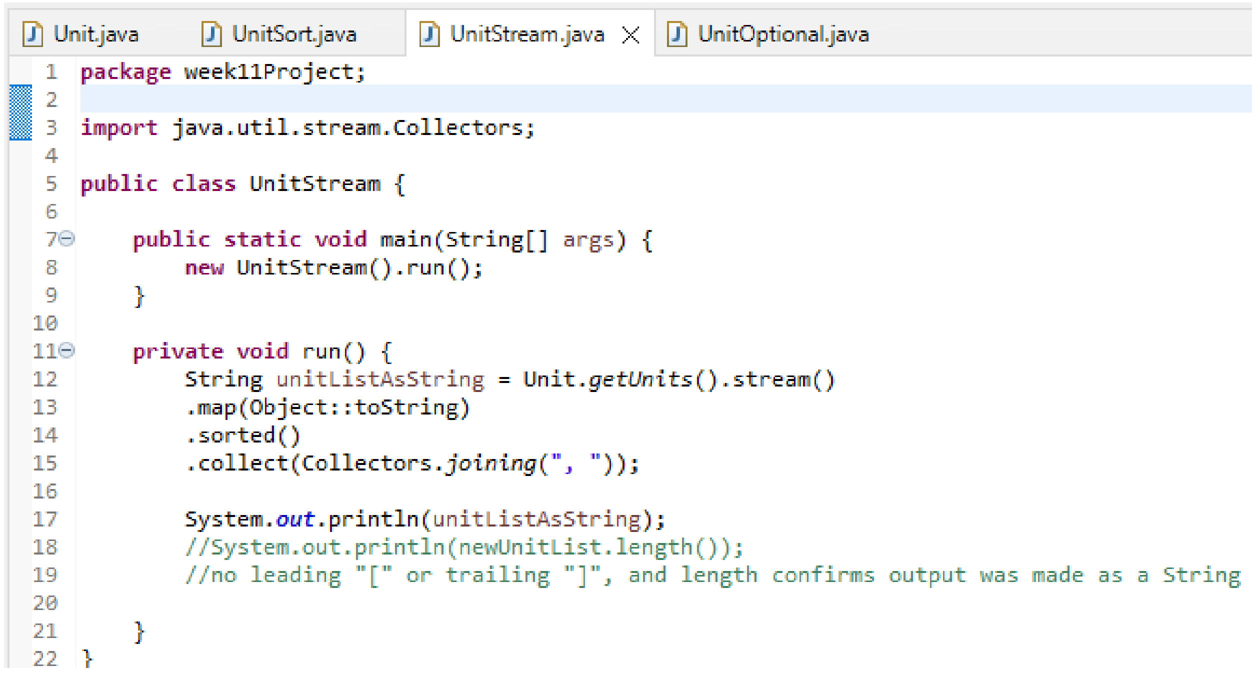
```java
package week11Project;

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

public class Unit {
	private String name;
	private String weapon;
	private String banner;
	private Boolean autoWin;

	//static data entry
	private static List<Unit> unitList = List.of(
	    new Unit("Yeoman", "Bow", "Dragon", Boolean.FALSE),
	    new Unit("Berserker", "Ax", "Bear", Boolean.FALSE),
	    new Unit("Acrobat", "Rod", "Giraffe", Boolean.FALSE),
	    new Unit("Grand Knight", "Imperial Edict", "Scorpion", Boolean.TRUE),
	    new Unit("Berserker's Brother", "Ax", "Bear", Boolean.FALSE)
	);

	//Class Constructor
	public Unit (String name, String weapon, String banner, Boolean autoWin) {
	    this.name = name;
	    this.weapon = weapon;
	    this.banner = banner;
	    this.autoWin = autoWin;
	}

```



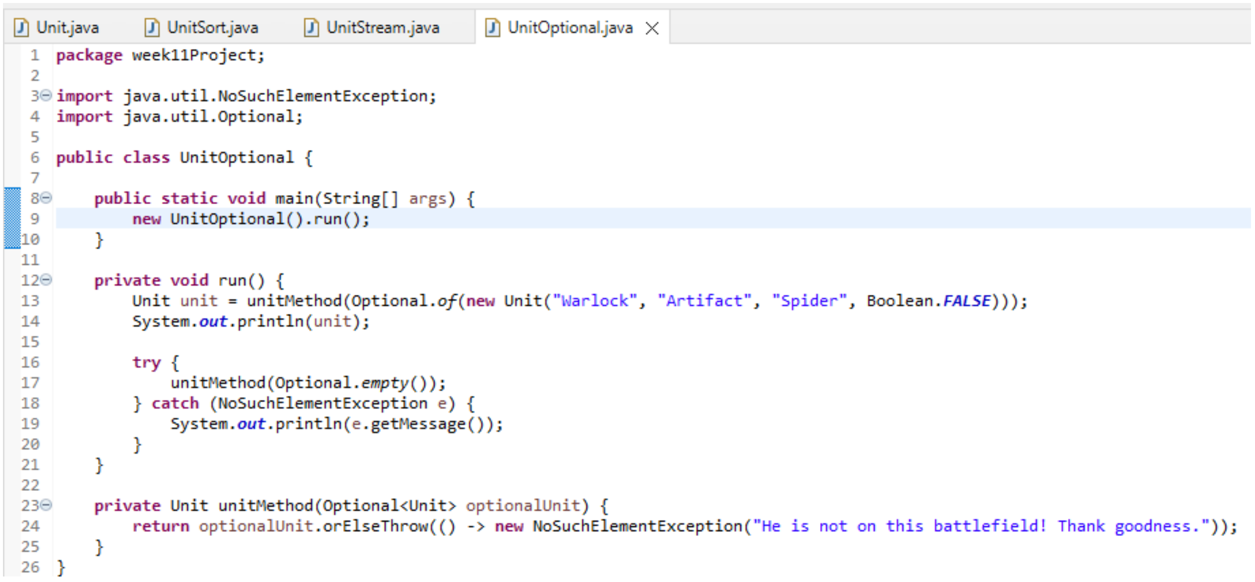
```java
        //Getters
        public String getUnit() {
            return name;
        }
        public String getWeapon() {
            return weapon;
        }
        public String getBanner() {
            return banner;
        }
        public Boolean isAutoWin() {
            return autoWin;
        }
        public static List<Unit> getUnits() {
            return new ArrayList<>(unitList); //Arraylist allows return list to be modifiable
        }

        @Override
        public String toString() {
            return banner + "'s " + name + " [" + weapon + "]";
        }

        public static int compare(Unit a, Unit b) {
            //AutoWin > Rod > Ax > Bow

            //A < B if:
                //1. A is not AutoWin and B is AutoWin
                //2. A is not AutoWin & Rod and B is Rod
                //3. A is Bow and B is not Bow (Ax or Rod)
            if  (  (!a.isAutoWin() && b.isAutoWin())
                || ((!a.isAutoWin() && !a.getWeapon().equals("Rod")) && b.getWeapon().equals("Rod"))
                || (a.getWeapon().equals("Bow") && !b.getWeapon().equals("Bow"))
                )
                return 1;

            //A = B
            if (a.getWeapon().equals(b.getWeapon()))
                return 0;

            //B > A (else)
            return -1;
        }
    }
```



```java
package week11Project;

import java.util.List;

public class UnitSort {

    public static void main(String[] args) {
        new UnitSort().run();
    }

    private void run() {
        List<Unit> lambdaUnits = sortByLambda();
        System.out.println(lambdaUnits);

        List<Unit> methodUnits = sortByMethod();
        System.out.println(methodUnits);
    }

    private List<Unit> sortByLambda() {
        List<Unit> units = Unit.getUnits();
        units.sort((u1, u2) -> Unit.compare(u1, u2));
        return units;
    }

    private List<Unit> sortByMethod() {
        List<Unit> units = Unit.getUnits();
        units.sort(Unit::compare);
        return units;
    }
}
```



```java
package week11Project;

import java.util.stream.Collectors;

public class UnitStream {

    public static void main(String[] args) {
        new UnitStream().run();
    }

    private void run() {
        String unitListAsString = Unit.getUnits().stream()
        .map(Object::toString)
        .sorted()
        .collect(Collectors.joining(", "));

        System.out.println(unitListAsString);
        //System.out.println(newUnitList.length());
        //no leading "[" or trailing "]", and length confirms output was made as a String

    }
}
```

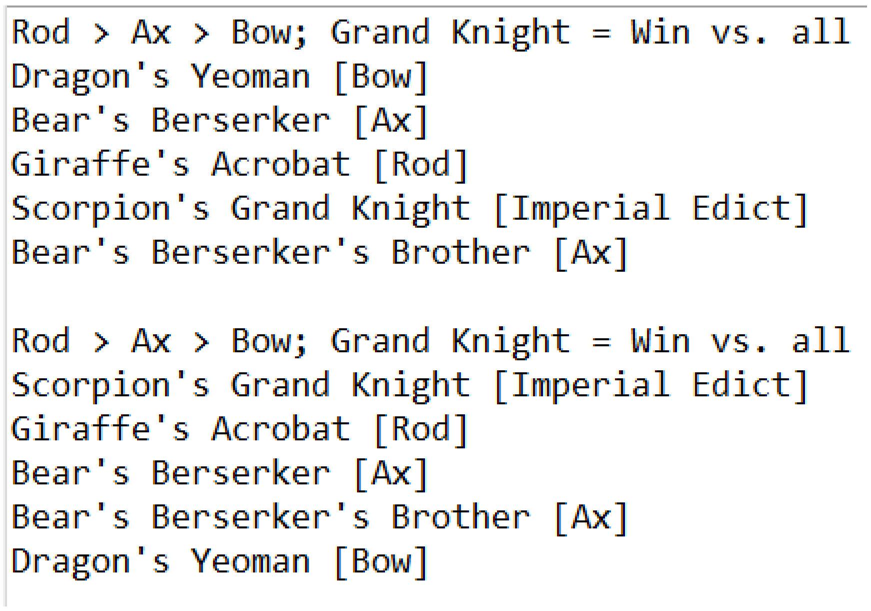


Unit.java | UnitSort.java | UnitStream.java | UnitOptional.java

```
package week11Project;

import java.util.NoSuchElementException;
import java.util.Optional;

public class UnitOptional {

    public static void main(String[] args) {
        new UnitOptional().run();
    }

    private void run() {
        Unit unit = unitMethod(Optional.of(new Unit("Warlock", "Artifact", "Spider", Boolean.FALSE)));
        System.out.println(unit);

        try {
            unitMethod(Optional.empty());
        } catch (NoSuchElementException e) {
            System.out.println(e.getMessage());
        }
    }

    private Unit unitMethod(Optional<Unit> optionalUnit) {
        return optionalUnit.orElseThrow(() -> new NoSuchElementException("He is not on this battlefield! Thank goodness."));
    }
}
```

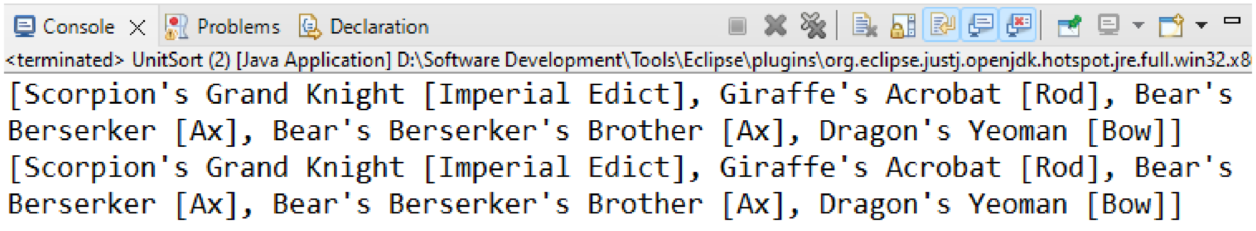
Screenshots of Running Application Results:

- Test Sorting:



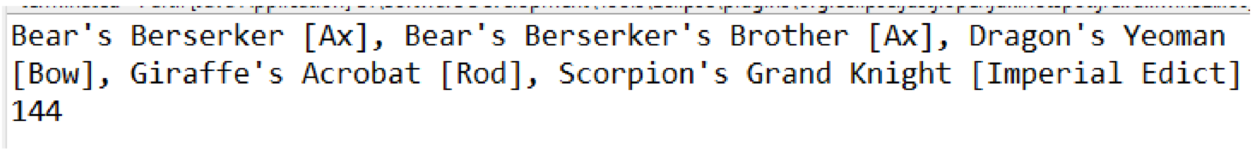
```
Rod > Ax > Bow; Grand Knight = Win vs. all
Dragon's Yeoman [Bow]
Bear's Berserker [Ax]
Giraffe's Acrobat [Rod]
Scorpion's Grand Knight [Imperial Edict]
Bear's Berserker's Brother [Ax]

Rod > Ax > Bow; Grand Knight = Win vs. all
Scorpion's Grand Knight [Imperial Edict]
Giraffe's Acrobat [Rod]
Bear's Berserker [Ax]
Bear's Berserker's Brother [Ax]
Dragon's Yeoman [Bow]
```

- Sorted lists output by Lambda expression and Method Reference:



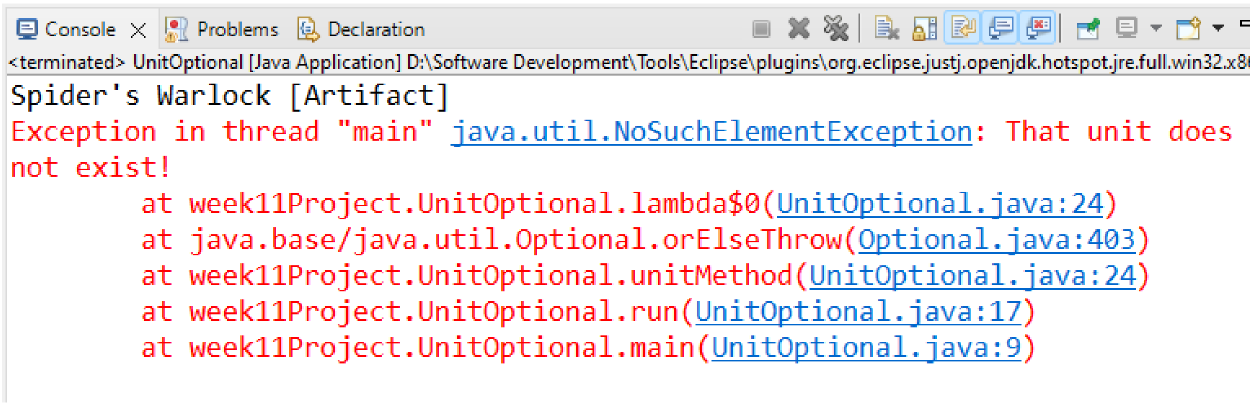
```
<terminated> UnitSort (2) [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
[Scorpion's Grand Knight [Imperial Edict], Giraffe's Acrobat [Rod], Bear's
Berserker [Ax], Bear's Berserker's Brother [Ax], Dragon's Yeoman [Bow]]
[Scorpion's Grand Knight [Imperial Edict], Giraffe's Acrobat [Rod], Bear's
Berserker [Ax], Bear's Berserker's Brother [Ax], Dragon's Yeoman [Bow]]
```

- Test Stream toString output (integer shown is String length to help confirm output is a single String)



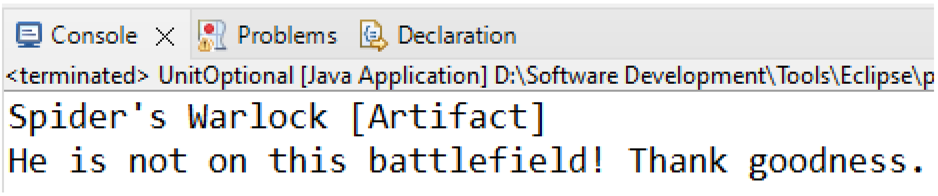
```
Bear's Berserker [Ax], Bear's Berserker's Brother [Ax], Dragon's Yeoman
[Bow], Giraffe's Acrobat [Rod], Scorpion's Grand Knight [Imperial Edict]
144
```

- Nonexistent optional before try/catch:



```
<terminated> UnitOptional [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
Spider's Warlock [Artifact]
Exception in thread "main" java.util.NoSuchElementException: That unit does
not exist!
	at week11Project.UnitOptional.lambda$0(UnitOptional.java:24)
	at java.base/java.util.Optional.orElseThrow(Optional.java:403)
	at week11Project.UnitOptional.unitMethod(UnitOptional.java:24)
	at week11Project.UnitOptional.run(UnitOptional.java:17)
	at week11Project.UnitOptional.main(UnitOptional.java:9)
```

- After coding of try/catch:



```
<terminated> UnitOptional [Java Application] D:\Software Development\Tools\Eclipse\p
Spider's Warlock [Artifact]
He is not on this battlefield! Thank goodness.
```

URL to GitHub Repository:

https://github.com/tylerjlivermore/Week11_MySQL_Lambdas_MethodRefs_Steams_Optionals