This is part two of a two-part final exam.

This part is open book. You may use any resources on a computer or paper except another person.

Answer the questions using eclipse. Zip up your completed project and name it firstname-lastname.zip (use your own first and last names). Submit it to the exam dropbox in before 6:35pm to the learning hub: Activities/Assignments/ExamPart2.

Name: _____ Score: _____ / 28

IMPORTANT: There are unit tests that mark this part of the exam. Follow the instructions precisely. The unit tests are included in the zip file along with this document.

Note: put all of your classes for this exam into a package called ca.bcit.comp1451.finalexam

1. Create a class which passes the following Junit tests (note: these are located on desire to learn for you to use):

```
package ca.bcit.comp1451.finalexam;
import static org.junit.Assert.*;
import org.junit.Test;
public class XTest {

	private X x;

	public XTest(){
		x = new X();
	}

	@Test
	public void C1(){
		try{
			x.setC("aaaaaaaaaa");
			fail();
		}catch(TooLongException e){
			assertEquals("aaaaaaaaaa is more than five characters so it is too long!", e.getMessage());
		}catch(Exception e){
			fail();
		}
	}

	@Test
	public void C2(){
		try{
			x.setC(".....");
			fail();
		}catch(TooLongException e){
			assertEquals("..... is more than five characters so it is too long!", e.getMessage());
		}catch(Exception e){
			fail();
		}
	}

	@Test
	public void C3(){
		try{
			x.setC(null);
			fail();
		}catch(TooShortException e){
```

```
                    assertEquals("null is not allowed; it is fewer than five characters so it is too short!",
                            e.getMessage());
            }catch(Exception e){
                    fail();
            }
    }

    @Test
    public void C4(){
            try{
                    x.setC("");
                    fail();
            }catch(TooShortException e){
                    assertEquals("empty string is fewer than five characters so it is too short!", e.getMessage());
            }catch(Exception e){
                    fail();
            }
    }

    @Test
    public void C5(){
            try{
                    x.setC("hi");
                    fail();
            }catch(TooShortException e){
                    assertEquals("hi is fewer than five characters so it is too short!", e.getMessage());
            }catch(Exception e){
                    fail();
            }
    }

    @Test
    public void C6(){
            try{
                    x.setC("99887");
                    assertEquals("99887", x.getC());
            }catch(Exception e){
                    fail();
            }
    }
    @Test
    public void C7(){
            assertTrue(x instanceOf W);
    }
}
```

2. Write a class called BooksWithNumbers which has a HashMap called "book titles". The HashMap values are Strings whose keys are Integers. Add the following keys and values into it in the BooksWithNumbers constructor, then use an Iterator of its keys to display all the HashMap values: (5 marks)

   1984 => "nineteen eighty four"
   2001 => "2001: a space odyssey"
   22 => "catch 22"
   451 => "farenheit: 451"

   Create a method called getTitle(int key) which returns the title whose key matches the parameter.

   Create a method called getNumberOfBooks() which returns the number of books (i.e. the number of elements in the map).

   Create a method addTitle() which takes a key and then a value as parameters and adds this to the map.

   Create a method called public String getTitles() which returns a String of all the titles put together in a row into a single String; for example:

   2001: a space odysseycatch 22farenheit: 451nineteen eighty four

   Create a method called public int getKeyFor(String title) which returns an int which is the key's value for the specified title; if there is no such title, instead throw an unchecked Exception type NoSuchTitleException.

3. Create a class hierarchy as follows:
   Parent class is Vehicle and is abstract
   Vehicle contains one instance variable; an int named weightPounds
   Vehicles are equal if they weigh the same amount; override equals
   Override toString so the object's class name and weight are returned
   Vehicles which weigh more are "bigger"; implement Comparable
   Class Car is a vehicle subclass; make it final
   Class Boat is a vehicle subclass; make it final too
   Class Vehicles has an ArrayList of Vehicles created, printed, sorted, and re-printed in its constructor:
   - A Car weighing 1000 pounds

- A Boat weighing 1200 pounds
- Another Car weighing 800 pounds
- Another Boat weight 900 pounds