IT 313 -- Final Exam

March 19, 2014

Part A: Short Answer Questions. Answer only 8 out of 11 questions. 5 points each.

1. What is the wrapper class for the primitive type `char`? For what are wrapper classes used?

2. Write a Java main method that inputs a string as a command line argument and prints to standard out the number of characters in this string.

3. _x is defined as this int instance variable:
```
private int _x;
```
This is the toString method:
```
@Override
public String toString( ) {
    return _x.toString( );
}
```
However, the following compiler error results:
```
Cannot invoke toString( ) on the primitive type int.
```
Fix the return statement so that no compiler error results.

4. What does `@Override` mean in Question 3?

5. The following code is supposed to compute the average of double values in a file:

```
Scanner in = new Scanner(new File("numbers.txt"))
double sum = 0.0;
int count = 0;
while(in.hasNextDouble( )) {
    double value = in.nextDouble( );
    sum += value;
    count++;
}
System.out.println("ave = " + sum / count);
```

However, when run, the program crashes with this error message:

```
Exception in thread "main" java.io.FileNotFoundException: numbers.txt (The
system cannot find the file specified)
```

Write a try/catch block that will prevent the program from crashing.

6. Here are the first five lines of the file `motor-boats.csv`:

```
vin, length, hp, price, manufacturer, model
SM4467", 17, 25, 30000, Bass Cat, Phelix
TE9382", 18,225, 36350, Reinell,  185LX
WQ3293", 29,350,129900, Sea Hunt, Game Fish
PT4231", 18, 90, 15000, Evinrude, E90SNL
```

The database table motorboats is created with the schema

```
create table motorboats(
    vin varchar(6),
    length integer,
    hp integer,
    price float,
    manufacturer varchar(15),
    model varchar(15)
);
```

This Java code reads from the files motor-boats.csv and inserts them into the table motorboats. Correct the errors. There are about 5 errors.

```
Scanner fromFile = new Scanner(new File("motor-boats.csv"));
fromFile.nextLine( );
while (fromFile.hasNextLine( )) {
    line = fromFile.nextLine( );
    fields = line.split(",");
    String vin = fields[0].trim( );
    String gender = fields[1].trim( );
    int length = Integer.parseInt(fields[2].trim( ));
    int hp = Integer.parseInt(fields[3].trim( ));
    String manufacturer = fields[4].trim( );
    String model = fields[4].trim( );
    sql = String.format(
        "insert into (vin, gender, length, hp, manufacturer, model)"+
        "values ('%s', '%s', %d, %d, '%s', '%s');",
            vin, gender, length, hp, manufacturer, model);
    System.out.println(sql);
    s.executeUpdate(sql2);
 }
```

7. What is the output when f(4) is called? Draw the recursion tree.

```
public static void f(int n) {
    if (n > 1) {
        System.out.print(n + 1 + " ");
        f(n - 3);
        System.out.print(n + " ");
        f(n - 1);
        System.out.print(n - 1 + " ");
    }
}
```

8. What changes must be made to the Person class on Page 5 so that the object sortable by age, so that the following code runs? Make the changes directly to the code on Page 5.

```
ArrayList<Person> p = new ArrayList<Person>(10);
p.add(new Person("Mason", 'M', 27));
p.add(new Person("Alice", 'F', 25));
p.add(new Person("Ruth", 'F', 30));
Collections.sort(p);
System.out.println(p);
```

9. If the Person class is defined as in Page 5 and the HashMap object map is defined as

```
HashMap<int, Person> map = new HashMap<int, Person>(50);
```

write a statement that adds to the hash map the Person object with fields
`_name == "Alice", _gender == 'F',` and **`_age == 25.`**

```
// Person class for Problems A9 and A10.
public class Person {

    private String _name;
    private char _gender;
    private int _age;

    public Person(String theName, char theGender, int theAge) {
        _name = theName;
        _gender = theGender;
        _age = theAge;
    }

    public String getName( ) {
        return _name;
    }

    public char getGender( ) {
        return _gender;
    }

    public int getAge( ) {
        return _age;
    }

    public void setAge(int theAge) {
        _age = theAge;
    }

    public void haveBirthday( ) {
        _age++;
    }

    @Override
    public String toString( ) {
        return _name + " " + _gender + " " + _age;
    }

}
```

10. Show the fill order for the FloodFill algorithm when `fill(3, 4)` is called?

```
public static void fill(int x, int y) {
    if (grid[y][x] != 0) {
        fill(x – 1, y);
        fill(x, y + 1);
        fill(x + 1, y);
        fill(x, y – 1);
    }
}
```

0	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1
1		1		1		1	1	1	1
1		1						1	1
1	1			*			1	1	1
1		1				1	1	1	1
1			1			1	1		1
1			1				1		1
1			1	1	1	1	1		1
1									1
1	1	1	1	1	1	1	1	1	1

11. What are some new languages that run on the Java Virtual Machine. How to they differ from Java?

Part B: Use Class from Java Class Library.

Select 4 methods from the Currency class in the Java Class Library—select 5 methods for extra credit. See the printed documentation for the Currency class. Write a main method that tests each of these methods.

Part C: Write Methods. Write code for these methods in the BoatStore Project. 5 points each.

Method Name	Class Name	Description	Page
setPrice	Vehicle	Setter for _price	8
toString	MotorBoat	Return all fields. Use call to super.toString.	9
remove	BasicInventory	Remove MotorBoat object with specified vin.	10
getSize	DerivedInventory	Return number of items in _col.	11
main	Test	Test BasicInventory class.	12

There are comments in this project that show where to enter the code for these methods.

```
==== Source code file Vehicle.java ============================

public class Vehicle {
     private String _vin;
     private int _length;
     private double _price;
     private int _horsePower;

     public Vehicle(String theVin, int theLength,
          int theHorsePower, double thePrice) {
          this._vin = theVin;
          this._length = theLength;
          this._horsePower = theHorsePower;
          this._price = thePrice;
     }

     public String getVin( ) {
         return _vin;
     }
     public int getLength( ) {
         return _length;
     }
     Public double getPrice( ) {

     }
     // Write code for setPrice, the setter for _price:
```

```
    public int getHorsePower( ) {
        return this._horsePower;
    }

    @Override
    public String toString( ) {
        return String.format("VIN: %s\n" +
                             "Length: %d\n" +
                             "Horse Power: %d\n" +
                             "Price: $%8.2f",
            _vin, _length, _horsePower, _price);
     }

}

==== Source code file MotorBoat.java ==========================

public class MotorBoat extends Vehicle {
    private String _manufacturer;
    private String _model;

    public MotorBoat(String theVin, int theLength,
        int theHorsePower, double thePrice,
        String theManufacturer, String theModel) {

        super(theVin, theLength, theHorsePower, thePrice);
        this._manufacturer = theManufacturer;
        this._model = theModel;
    }

    // Write toString method here.  You can call
    // super.toString if you wish.









}
```

```
==== Source code file BasicInventory.java =====================

import java.util.ArrayList;
public class BasicInventory {
    public ArrayList<MotorBoat> _col;
    public BasicInventory(int theCapacity) {
        _col = new ArrayList<MotorBoat>(theCapacity);
    }

    public void add(MotorBoat theBoat) {
        _col.add(theBoat);
    }

    public MotorBoat display(String theVin) {
        for(MotorBoat b : _col) {
            if(b.getVin( ).equals(theVin)) {
                return b;
            }
        }
        return null;
    }

    // Write code for remove here.  You can use some of the
    // code in the preceding display method.




    @Override
    public String toString( ) {
        String retVal = "";
        for(MotorBoat b : _col) {
            retVal += b.toString( );
        }
        return retVal;
    }
}
```

```
==== Source code file DerivedInventory.java ===================

    public class DerivedInventory extends BasicInventory{
        public DerivedInventory(int theCapacity) {
            super(theCapacity);
        }

        // Write code for getSize method here.







    }
```

```
==== Source code file Test.java ================================

public class Test {
    public static void main(String[] args) {
        MotorBoat a = new MotorBoat("SM4467", 17, 25, 30000,
            "Bass Cat", "Phelix");
        MotorBoat b = new MotorBoat("TE9382", 18, 225, 36350,
            "Reinell", "185LX");
        MotorBoat c = new MotorBoat("WQ3293", 29, 350, 129900,
            "Sea Hunt", "Game Fish");
        MotorBoat d = new MotorBoat("PT4231", 18, 90, 15000,
            "Evinrude", "E90SNL");

        DerivedInventory di = new DerivedInventory(20);

        // Write code to
        // (1) insert MotorBoat objects a, b, c, and d into di.
        // (2) display the object with vin "WQ3293"
        // (3) print the return value of the getSize method.
        // (4) use the toString method of di to show all of the
        //         objects in di.







    }
}
```