IT 313 – Advanced Application Development

Practice Midterm Exam

Part A. Multiple Choice Questions. Answer all questions. Optional: supply a reason or show work for partial credit in case you select the wrong answer. If you select the correct answer, your reason of work will not be considered. 5 points for each question.

1. Which statement checks whether the String objects `s` and `t` contain the same characters?

   a. `equals(s, t)`   b. `s.equals(t)`   c. `s = t`   d. `s == t`

2. Which of the following does NOT denote a Java comment?

   a. `// This is a comment.`

   b. `/* This is a comment. */`

   c.
   ```
   =begin
   This is a comment.
   =end
   ```
   d.
   ```
   /**
    *  This is a comment.
    */
   ```

3. The ordered list of parameter datatypes for a method is called the method __________ .

   a. instance variables list   b. local variables list   c. post list   d. signature

4. Which statement converts the int value `n` into the String object `t`?

   a. `t = n.toString( );`   b. `t = String.parseString(n);`

   c. `t = (String) n;`   d. `t = String.valueOf(n);`

5. Which statement converts the char value `c` into its ASCII code n?

   a. `n = c.toAscii( );`   b. `n = (int) c;`

   c. `n = (char) c;`   d. `n = Integer.valueOf(c);`

6. Which of these is the best definition for the setter for the **`_price`** instance variable defined by
   `private double _price;`

   a.
   ```
   public double set_price(double thePrice) {
         _price == thePrice;
   }
   ```
   b.
   ```
   public setPrice(double thePrice)double {
         _price = thePrice;
   }
   ```
   c.
   ```
   public setPrice(double thePrice)double
         return _price = thePrice;
   ```
   d.
   ```
   public void setPrice(double thePrice) {
         _price = thePrice;
   }
   ```

7. The collection `col` is defined as
   `ArrayList<String> col = new ArrayList<String>( );`
   Which statement obtains the leftmost character of the string at index 3?
   For example, the leftmost character of `"chicago"` is `'c'`.

   a. `char c = col.get(3).charAt(0)`
   b. `char c = col.get(3).charAt(1)`
   c. `char c = col.index(3).charAt(0)`
   d. `char c = col.index(3).charAt(1)`

8. In a `HashMap` collection, the percentage of occupied cells is called the _______________ .
   a. efficiency b. load factor c. redundancy d. throughput

9. The `BankAccount` interface requires `deposit` and `withdraw` methods. Which of the following is a syntactically and logically correct definition of `BankAccount`?

   a.
```
public interface BankAccount {
     public boolean deposit(double amount) { }
     public boolean withdraw(double amount) { }
}
```

   b.
```
public interface BankAccount {
     public boolean withdraw(double amount) { }
}
```

   c.
```
public interface BankAccount {
     public boolean deposit(double amount);
}
```

   d.
```
public interface BankAccount {
     public boolean deposit(double amount);
     public boolean withdraw(double amount);
}
```

10. A class that implements the `Comparable` interface must supply a definition of the _____________ method.
    a. `equals` b. `compareTo` c. `sort` d. `spaceship`

Part B: Write a main Method. 15 points. The input file `salaries.txt` contains information about company executives. Here are the first three lines of the input file:

```
    Rory McFaddin,M,Chicago,380000
    Justine Douglas,F,Los Angeles,490000
    Alice Chen,F,Chicago,390000
```

The fields are name, gender, city, and salary.

The following `main` method reads from the input file and writes a line to the output file like this:

```
    Average salary, for gender F and city Chicago: $??????.
```

(It writes the actual average salary of female executives instead of ??????.) In the `while` loop, use an if statement to update the count and the sum of the salaries for only the lines where gender is `'F'` and city is `"Chicago"`.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Main {
     public static void main(String[] args) throws FileNotFoundException {
          Scanner s = new Scanner(new File("salaries.txt"));
          double sum = 0.0;
          int count = 0;
          while(s.hasNextLine( )) {
               String[ ] fields = s.nextLine( ).split(",");
               // Add your code here ----------------------------------









               // ---------------------------------------------------------
          }
          s.close( );

          PrintWriter pw = new PrintWriter("result.txt");
          pw.println("Average salary, for gender F and city Chicago: " +
              sum / count);
          pw.close( );
     }
}
```

Part C: Correct the Errors. There are about five errors in each of the three files `Pet.java` (this page), **`Dog.java`** (Page 5), and **`Main.java`** (Page 5). Correct these errors. Inserting, correcting, or removing a pair of `' '`, `" "`, `( )`, `[ ]`, or `{ }` only counts as one error. 15 points.

```
//------- Source code file: Pet.java

package it313.finalexam.petsalon {
    public class Pet {
        private String _name;
        private String _owner;
        private int _age;

        public Pet(String theName, String theOwner, int theAge) {
            _name = theName;
            _owner = theOwner;
            theAge = _age;
        }

        public String getName {
            return _name;
        }

        public String getOwner( ) {
            return _owner;
        }

        public void getAge( ) {
            return _age;

        @Overrides
        public String to_string( ) {
            return String.format("%s %s %d", _name, _owner, _age);
        }
    }
}
```

```
//------- Source code file: Dog.java

package it313.finalexam.petsalon;
public class Dog super Pet {
    private String _breed;

    public void Dog(String theName, String theOwner, String theAge,
               String theBreed) {
        super(theName, theOwner);
        _breed = theBreed;

    }

    public string getBreed() {
        return _breed;
    }

    @Overrides
    public String toString( ) {
        return super.toString( ) + String.format(" %d", _breed);
    }
}


//------- Source code file: Test.java

public class Test {

    public static void main(String args) {

        ArrayList col = new ArrayList( )
        col.add(new Dog("Vini", "Jack", 2, "Pomeranian"));
        col.add(new Dog("Klaus", "Nancy", 4, "German Shephard"));
        col.add(new Dog("Sparky", "Steve", 1, Dachshund));

        for(Dog d in col) {
           System.out.println(d);
        }

        System.out.println( );
        System.out.println(col[2].toString( ));
    }
}
```

Part D: Test Methods from Java Class Library. Documentation from the Java Class Library for the `Stack` class is shown on Page 7. Write traditional tests for three `Stack` instance methods. 10 points. For 3 points extra credit (13 total points), instead write a Junit test script to test the three `Stack` instance methods. If you complete both the traditional tests and the JUnit tests, I will throw out the worst one.

A `Stack` is like an `ArrayList` collection, except that you can only add an item to the top of the array list (using `push`) or remove an item from the top of the array list (using `pop`). Some other instance methods of the `Stack` class are `peek`, `empty`, and `search`.

```
=== Traditional tests ========================

public static void main(String[ ] args) {
    Stack<String> s = new Stack<String>( );







}

=== Junit tests ==============================

import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;

public class MainTest {
     private Stack<String> s;

     @Before
     public void setup( ) {
         s = new Stack<String>( );
     }

     @Test
     public void test( ) {











     }
}
```

Java Class Library Documentation for the Stack Class

Constructor and Description

`Stack<E>( )`
Creates an empty **`Stack`** object that contains objects of datatype **`E`**.

Some Instance Methods and Descriptions

`boolean empty( )`
Returns true if this stack is empty, false otherwise.

`E peek( )`
Looks at the object at the top of this stack without removing it from the stack.

`E pop( )`
Removes the object at the top of this stack and returns that object.

`E push( )`
Pushes an item onto the top of this stack and also returns the item.

`int search(E item)`
Returns the 1-based position where an item is on this stack.