This is a closed-book test; no aids are allowed.

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

You may write in pen or pencil.

The test is 80 minutes in duration.

Name and Student Number: _____

1.   / 35

2.   / 30

3.   / 25

4.   / 25

5.   / 10

6.   / 25

      / 150

1. Types.

   (a) (5 points) `int` and `double` are different types in Java. In one or two sentences, explain what the word "type" means in a programming language such as Java.

      > **Solution:** The set of values that a variable can take, and the operations that can be performed using the variable.

   (b) (5 points) The + operator in Java is called an overloaded operator because it can be used for more than one kind of operation. How does Java determine what operation to perform for an expression like `(x + y)`?

      > **Solution:** It looks at the types of `x` and `y`.

   (c) (5 points) Suppose that `x` has type `int` and `y` has type `double`. What type does the expression `(x + y)` have?

      > **Solution:** `double`

   (d) (5 points) Suppose that the following statement compiles successfully in a Java program:

      ```
      double a = (b + c);
      ```

      What type or types can the expression `(b + c)` have?

      > **Solution:** Any primitive numeric type.

   (e) (5 points) The textbook and the lecture slides both state that Java is a strongly typed language. In one sentence, explain what "strongly typed" means in a programming language such as Java.

      > **Solution:** Every variable must be declared (with a type) before it can be used.

   (f) (10 points) Consider your answer to Question 1(e): Why is closure an important property of the primitive types in Java?

      > **Solution:** Without closure, it is impossible to determine at compile time what the type is of an expression such as `x + y` (where `x` and `y` are primitive numeric types). This means it is impossible to determine what type of variable can hold the result of `x + y`, which is a problem in strongly typed languages.

2. For parts (a)–(c) of this question, consider the contract for the method `daysInYear` in the utility class named `PlanetEarth`; the class `PlanetEarth` gathers information about our home planet:

```
static int daysInYear(long year)
```

Returns the number of days in the given calendar year.

Precondition:
    year >= 0
Parameters:
    year - the year
Postcondition:
    returns the number of days in year

(a) (5 points) In one sentence explain why it makes sense that `PlanetEarth` is a utility class.

> **Solution:** There is only one planet earth.

(b) (5 points) Suppose that as a client, you write the following two lines of code

```
int days = PlanetEarth.daysInYear(-2);
System.out.println(days);
```

which prints out the number `-9999`. Using the contract analogy for methods, who (the client or the method?) is responsible for the unusual output, and why are they responsible?

> **Solution:** The client is responsible because the precondition was not true.

(c) (5 points) Suppose that as a client, you write the following two lines of code

```
int days = PlanetEarth.daysInYear(2011);
System.out.println(days);
```

which prints out the number 100. Using the contract analogy for methods, who (the client or the method?) is responsible for the unusual output, and why are they responsible?

> **Solution:** The method is responsible because the precondition was true and the postcondition was false.

(d) (5 points) Another student says to you, "The method `daysInYear` is silly. There should be a public attribute named `PlanetEarth.DAYS_IN_YEAR` instead." Explain in one or two sentences why you agree or disagree with the other student.

> **Solution:** Disagree. A `public final` attribute should not be used because the number of days in a year is not a constant (consider leap years for example).

(e) (10 points) The textbook says that public attributes in a class are risky, but the utility class `java.lang.Math` has two public attributes named `E` and `PI`!

Why are public attributes risky, and why are the attributes in `java.lang.Math` not risky?

> **Solution:** The `java.lang.Math` public attributes are not risky because they are primitive types and they are marked `final`. This means that the values of `Math.E` and `Math.PI` cannot be changed by the client.

3. APIs and methods

   Here are eight method summaries from the API for `java.lang.String`:

   1. `boolean equals(Object anObject)`
      Compares this string to the specified object.

   2. `int indexOf(int ch)`
      Returns the index within this string of the first occurrence of the specified character.

   3. `int indexOf(int ch, int fromIndex)`
      Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

   4. `int indexOf(String str)`
      Returns the index within this string of the first occurrence of the specified substring.

   5. `int indexOf(String str, int fromIndex)`
      Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

   6. `String toString()`
      This object (which is already a string!) is itself returned.

   7. `static String valueOf(int i)`
      Returns the string representation of the `int` argument.

   8. `static String valueOf(double d)`
      Returns the string representation of the `double` argument.

   (a) (5 points) Beside each line of code below, fill in the blanks with the number corresponding to which method is invoked; use the word "none" if you think that none of the eight listed methods is invoked. Assume that `s` and `t` are both `String` references.

   ```
   s.indexOf('a', 5);              _____3_____

   s.indexOf("a", 5);              _____4_____

   s == t;                         ___none____

   s.equals(t);                    _____1_____

   String.valueOf(1 / 3);          _____7_____
   ```

(b) (5 points) Beside each method number below, fill in the blank corresponding to the type returned by the method:

```
method 1      __boolean__

method 4      ____int____

method 5      ____int____

method 6      ___String__

method 7      ___String__
```

(c) (5 points) Which of the eight methods, if any, are considered to be obligatory methods (methods that every class must have)?

> **Solution:** 1 and 6

(d) (10 points) Explain what pass-by-value means in Java, and explain how pass-by-value can help promote software reliability.

> **Solution:** When arguments are passed to a method, the method receives copies of the values of the arguments. This promotes software reliability because as clients, we have the assurance that methods cannot alter the values of our variables.

4. Consider the class `type.lib.Fraction` for each part of this question. Assume that each memory diagram that you are asked to draw represents a different Java program (e.g., the addresses you use in part (a) can be the same as the ones you use in part (b)).

(a) (5 points) Draw a memory diagram for the following two lines of code; you do not need to show the `Fraction` class in memory, but you should show the values of any numerators and denominators.

```
Fraction f = new Fraction(1, 5);
Fraction g = f;
```

⋮	
100	**main** invocation
f	600
g	600
⋮	
600	**Fraction** object
denominator	1
numerator	5
⋮	

(b) (5 points) Draw a memory diagram for the following two lines of code; you do not need to show the `Fraction` class in memory, but you should show the values of any numerators and denominators.

```
Fraction f = new Fraction(1, 5);
Fraction g = new Fraction(1, 5);
```

⋮	
100	**main** invocation
f	600
g	700
⋮	
600	**Fraction** object
denominator	1
numerator	5
⋮	
700	**Fraction** object
denominator	1
numerator	5
⋮	

(c) (5 points) Draw a memory diagram for the following lines of code; you **must** show the `Fraction` class in memory, and you should show the values of any numerators and denominators.

```
Fraction f = new Fraction(1, 5);
Fraction.isQuoted = false;
```

⋮	
100	`main` invocation
f	600
⋮	
500	`Fraction` class
denominator	
numerator	
isQuoted	false
⋮	
600	`Fraction` object
denominator	1
numerator	5
⋮	

(d) (5 points) The textbook says that an object has both identity and state. In 4(c), what is the identity of the `Fraction` object referenced by `f`?

> **Solution:** The memory address where the object is located, i.e., 600.

(e) (5 points) The textbook says that an object has both identity and state. In 4(c), what is the state of the `Fraction` object referenced by `f`?

> **Solution:** The values of its attributes, i.e., 1 and 5.

5. (10 points) Suppose that you have two `ArrayList<Integer>` objects x1 and x2 both having
   size equal to four, and containing the following elements:

   ```
   x1:  [0, 1, 2, 3]
   x2:  [1, 2, 3, 3]
   ```

   Consider the following fragment of Java code:

   ```java
   int i1 = 0;
   int i2 = 0;
   int count = 0;
   while ( (i1 < x1.size()) && (i2 < x2.size()))
   {
      // A
      int elem1 = x1.get(i1);
      int elem2 = x2.get(i2);
      if (elem1 == elem2)
      {
         count++;
         i2++;
      }
      else if (elem1 < elem2)
      {
         i1++;
      }
      else
      {
         i2++;
      }
      // B
   }
   ```

   What is the value of `count` after this code has finished running? *Hint: Consider tracing the
   values of i1, i2, and count at points A and B each time through the loop.*

   > **Solution:** Run the program, printing out the values of `i1` and `i1` at points `A` and `B.Math.E`

6. Strings. All parts of this question are about Java strings (as they are implemented in `java.lang.String`).

    (a) (5 points) The `String` class has no mutator methods; classes that have no mutator methods are said to be _____.

    > **Solution:** immutable

    (b) (5 points) The string operation performed by the `+` operator is called _____.

    > **Solution:** concatenation

    (c) (5 points) Consider the following fragment of Java code:

    ```
    String s = "Four : 4";
    String t = "Four : " + "4";
    String t = "Four : " + 4;
    String v = "Four : " + 2 + 2;
    ```

    One of the strings represents a different sequence of characters than the other three strings. Explain why.

    > **Solution:** Each expression on the right hand side of an assignment involves only concatenation; thus, the expressions are evaluated from left to right which means `v` represents the string `"Four:  22"`

    (d) (5 points) Given two string references `s` and `t`, how would you check if they represent different sequences of characters?

    > **Solution:** Use `!` and `equals`
    > ```
    > if (!s.equals(t))
    > {
    > }
    > ```

    (e) (5 points) All courses in Computer Science and Engineering have a course code that begins with the string `CSE` followed by four digits where the first digit cannot be zero, eight, or nine. Write down a regular expression that would match as closely as possible all CSE course codes.

    > **Solution:** `"CSE[1-7][0-9][0-9][0-9]"`