EXERCISES FOR ROLE PLAYING FROM THE UNIVERSITY OF WISCONSIN-MADISON

Exercise 1: Java Identifiers and Naming Conventions

What are the different kinds of things in a Java program that get names?

For each of the following,

- Say whether it's a valid Java name (identifier).
- If it is, does it follow the naming conventions discussed in class?
- If it does follow the naming conventions, what kinds of names should it be used for?
- If it doesn't follow the naming conventions, how could you fix it?

xyz	this Is Fine
WillThisWork?	public
letterCount	Public
fire-fly	number$1
MAX_VAL	Sandwich
mustang	abiggreenpickle
true	go#home
3people	big_bear

Exercise 2: Java variables and types

Remember that in Java

- variables must be declared and initialized before they are used,
- operators (like +, –, and so on) must be applied only to variables and literals of the correct type,
- in general, the types of the left and right-hand sides of an assignment must match (but note that an *int* value can be assigned to a *double* variable, but not vice-versa),
- the types of the arguments in a method call must match the types of the corresponding parameters, and the value returned must match the method's return type (again, an *int* can be used where a *double* is expected, but not vice-versa).

Keeping all this in mind, find all of the errors that the Java compiler would report in the following code.

```
public int doDivision( double denom ) {
    int returnVal;
    double num1 = 4.4;
    num2 = 5.5;
    returnVal = (num1 + num2)/denom;
    return returnVal;
}

public int computeVal( int oneVal ) {
    int returnVal;
    int i1, i2, i3;
    Integer intObj;
    i1 = 22222222222222;
    intObj = 11;
    i2 = num1;
    i3 = doDivision( "hello" );
    returnVal = (i1 + i2)/10000.0;
    Return returnVal;
}
```

Exercise 3: Arithmetic expressions

Translate the following formulas to Java expressions, using methods from the *Math* class where appropriate. Assume that all variables are of type *double*.

(a) $\dfrac{n\ (n + 1)\ (2n + 1)}{6}$

(b) $a^2 + b^2 + 2ab\ cos(c)$

(c) $2\cos^2(a) - 1$

(d) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$

(e) $ut + 1/2\ at^2$

Some Math Class Methods for Double Values

`static double`	`abs(double a)` Returns the absolute value of a `double` value.
`static double`	`cbrt(double a)` Returns the cube root of a double value.
`static double`	`cos(double a)` Returns the trigonometric cosine of an angle.
`static double`	`max(double a, double b)` Returns the greater of two `double` values.
`static double`	`min(double a, double b)` Returns the smaller of two `double` values.
`static double`	`pow(double a, double b)` Returns the value of the first argument raised to the power of the second argument.
`static double`	`sin(double a)` Returns the trigonometric sine of an angle.
`static double`	`sqrt(double a)` Returns the correctly rounded positive square root of a `double` value.
`static double`	`tan(double a)` Returns the trigonometric tangent of an angle.

Exercise 4: Class methods and data vs Instance methods and data

This exercise will help you to understand the difference between class and instance fields, between class and instance methods, and between public and private methods.

Part (a) Look at the *Book* class defined below. Say which fields are *class* fields and which are *instance* fields. Do the same for the methods.

```
public class Book {
   private String title;
   private double price;
   private static int totalNumBooks;
   private String author;
   private int numSold;
   public static final double MIN_PRICE = 2.0;

   public Book(String aTitle, double aPrice, String anAuthor) { ... }

   public double getPrice( ) { ... }

   private void lowerPrice( ) { ... }

   private static double newPrice(double oldPrice, int discount) { ... }
}
```

Part (b) Assume that the following (nonsense) code is in a method that is in the *Book* class. Find and fix all of the errors.

```
      Book oneBook = new Book("Harry Potter", 19.95, "Rowling");
      MIN_PRICE = oneBook.getPrice();
      oneBook.lowerPrice();
      double price = Book.getPrice();
      System.out.println(oneBook.numSold);
      System.out.println(MIN_PRICE);
```

Part (c) Now assume that the code given above is *not* in the *Book* class. What new errors arise?

Exercise 5: Practice with Interfaces

This question uses the following two interface definitions.

```
public interface Person {
    String getName();
}

public interface Employee {
    double getSalary();
}
```

Part (a)

None of the following classes correctly implement the *Person* and/or the *Employee* interface. Find and fix all of the errors.

```
public class Child implements Person {
    private String name;

    private String getName() { return name; }
}

public class Adult implements Person, Employee {
    private String name;
    private double salary;

    public String getName() { return name; }
}

public class Worker implements Employee {
    private double salary;

    public String getSalary() { return salary; }
}
```

Part (b)

Below is code that correctly defines two classes that implement the *Person* and *Employee* interfaces.

```
public class Student implements Person {
    private String name;
    private int age;
    public Student(String aName, int anAge) {
        name = aName;
        age = anAge;
    }
    public String getName() { return name; }
    public int getAge() { return age; }
}

public class Teacher implements Person, Employee {
    private String name;
    private double salary;
    public Teacher(String aName, double sal) {
        name = aName;
        salary = sal;
    }
    public String getName() { return "Professor " + name; }
    public double getSalary() { return salary; }
}
```

The code in the *main* method of the *Test* class defined below will not compile. Find and explain all of the errors.

```
public class Test {
    public static void printName(Person pers) {
        System.out.println(pers.getName());
    }
    public static void main(String[] args) {
        Person onePerson = null;
        Employee oneEmployee = null;
        Student oneStudent = null;
        Teacher oneTeacher = null;
        onePerson = new Person("Sandy", 12);
        onePerson = new Teacher("Sandy", 12);
        oneEmployee = oneStudent;
        oneStudent = onePerson;
        onePerson = oneStudent;
        int age = onePerson.getAge();
        printName(onePerson);
        printName(oneStudent);
        printName(oneTeacher);
        printName(oneEmployee);
    }
}
```

Part (c)

Suppose that you replace the bad code in the *main* method above with the following code. What is printed when this code executes?

```
onePerson = new Student("Sandy", 12);
printName(onePerson);
onePerson = new Teacher("Sandy", 12);
printName(onePerson);
```