

First Name: \_\_\_\_\_

AITI 2004: Exam 2 – July 19, 2004

Last Name: \_\_\_\_\_

Standard Track

**Read Instructions Carefully!**

This is a 3 hour closed book exam. No calculators are allowed. Please write clearly – if we cannot understand your answer, you will receive 0 points for that question. This exam has 6 parts worth a total of 100 points.

**Section 1 – Abstract Classes and Interfaces (35 points)**

1. (14 points) Circle True or False

True / False If a class uses an interface, we use the keyword `extends`.

True / False An interface contains a list of methods that every class which implements this interface must have.

True / False An abstract class can be instantiated.

True / False If there exists an interface called `Animal`, the following statement will compile successfully:

```
Animal a1 = new Animal();
```

True / False If there exists an interface called `Animal`, and a class called `Lizard`, the following statement will compile successfully:

```
Animal a1 = new Lizard();
```

True / False You can implement methods in the body of an abstract class.

True / False A class can be the direct subclass of several classes.

True / False A class can implement several interfaces.

2. (6 points) There are three Java files between the horizontal lines on the next page: `Shape.java`, `Rectangle.java`, and `Square.java`. Each is in a separate package. Add `import` statements to the files so that they compile.

3. (3 points) Implement the `Square` constructor on page 3.

4. (12 points) Looking at the classes on page 3, circle the output of the following code segments or circle "error" if the code would generate a compiling or execution error.

```
Square s = new Square(3);  
System.out.println(s.area());
```

a) 3                      b) 9                      c) error

```
Rectangle r = new Rectangle(3, 3);  
System.out.println(r.area());
```

a) 3                      b) 9                      c) error

```
Shape p = new Shape(3, 3);  
System.out.println(p.area());
```

LAST NAME, FIRST NAME:

a) 3                    b) 9                    c) error

```
Square s = new Square(3);  
System.out.println(s.toString());  
a) square            b) rectangle    c) error
```

```
Rectangle r = new Square(3);  
System.out.println(r.toString());  
a) square            b) rectangle    c) error
```

```
Shape p = new Square(3);  
System.out.println(p.toString());  
a) square            b) rectangle    c) error
```

```
package shapes;
```

```
public interface Shape {  
    public int area();  
}
```

---

```
package rectangles;
```

```
public abstract class Rectangle implements Shape {  
    private int width, height;  
  
    public Rectangle(int width, int height) {  
        this.width = width;  
        this.height = height;  
    }  
  
    public int area() { return width * height; }  
  
    public String toString() { return "rectangle"; }  
}
```

---

```
package squares;
```

```
public class Square extends Rectangle implements Shape {  
    public Square(int size) {  
  
    }  
  
    public String toString() { return "square"; }  
}
```

## Section 2 – Exceptions (12 points)

1. (4 points) Create a checked exception class called `DivideByZeroException`.

2. (4 points) Change this method so that it throws a `DivideByZeroException` with an appropriate message if `b` is zero.

```
public static double divide(double a, double b) {  
  
  
  
  
  
  
  
  
  
    return a / b;  
}
```

3. (4 points) Change this method so that it catches the `DivideByZeroException` thrown by `divide` and prints the message "the divisor is zero" if it is thrown.

```
public static void printQuotient(double a, double b) {  
  
  
  
  
  
  
  
  
  
    System.out.println(divide(a, b));  
  
  
  
  
  
  
  
  
  
}
```

### Section 3 – There is NO Section 3

J

### Section 4 – IO and Parsing (12 points)

1. (2 points) In order to read/write files in Java, you should import which package?

- a) `javax.swing`
- b) `java.io`
- c) `java.awt`
- d) `java.util`

2. (2 points) To write to a byte stream, you should use a class that extends which of the following abstract classes?

- a) `Writer`
- b) `Reader`
- c) `InputStream`
- d) `OutputStream`

3. (2 points) To read from a character stream, you should use a class that extends which of the following abstract classes?

- a) `Writer`
- b) `Reader`
- c) `InputStream`
- d) `OutputStream`

4. (6 points) Complete the method `prodDoub`s below. It expects a `String` argument containing doubles separated by `!` symbols and should return the sum of those integers. For example, when passed the `String "2.5!1.0!8.1"`, it should return `20.25`. You can assume `java.util.*` has been imported.

```
public static double prodDoub(String doubString) {
```

```
}
```

LAST NAME, FIRST NAME:

## Section 5 – Swing (21 points)

1. (2 points) In order to create graphical interfaces in Java, you should import which packages?

- a) `java.awt.event, javax.swing`
- b) `java.util`
- c) `java.util, javax.swing`
- d) `java.io, java.swing`

2. (2 points) What does the acronym GUI stand for?

3. (2 points) What are the pixel coordinates of the top left corner of your screen?

4. (3 points) Briefly describe how the Flow Layout Manager lays out its components.

5. (12 points) Match the following Swing component with what it is used for. You may write the letter of the answer in the line next to the component:

<b>Component</b>	<b>Use</b>
1. ____ <code>JTextField</code>	A) waits for the user to click on it
2. ____ <code>JComponent</code>	B) A top level container
3. ____ <code>JButton</code>	C) Provides space for the user to type into
4. ____ <code>JPanel</code>	D) All the Swing components inherit from this class
5. ____ <code>JFrame</code>	E) An intermediate container
6. ____ <code>JLabel</code>	F) Displays text that the user cannot edit

LAST NAME, FIRST NAME:

## Section 6 – Classes (20 points)

On the next page write a class to represent a football team in a package called `football`. We want to be able to access the class `FootballTeam` from **any** package.

The team has the following properties, which **cannot** be accessed outside of the class.

- name of the team
- number of wins
- number of losses

Write a constructor that accepts the name of the team, the number of wins, and the number of losses as arguments and sets the class properties to those values. This constructor should be accessible from **any** package.

Write a second constructor that takes only the name of the team as an argument. This constructor should set the name of the team to the argument and set the number of wins and losses to 0. (Hint: The body of this constructor should be one line and it should call the first constructor.) This constructor should be accessible from **any** package.

Write methods that return the name of the team, the number of the wins, and the number of losses. These methods should be accessible from **any** package.

Next write a method to increase the numbers of wins by 1 and another method to increase the number of losses by one. These methods should only be accessible from **inside** the `football` package.

Write a method that returns true when a team has a "good" record, meaning the team has more wins than losses. This method should be accessible from **any** package.

Finally, add a `main` method to the `FootballTeam` class. In the `main` method, construct a `FootballTeam` named "AITI" with 3 wins and 5 losses. Call the method that returns true when the team has a good record and print out the result. Now make three calls to the method that increases the number of wins by 1. Lastly, call the "good record" method again and print out the result.