What is your name?: [Key] (0 points)

There are two sections:
I. True/False 40 points; (2 points each, 20 questions)
II. Problems 60 points; (6 points each, 10 questions)

100 points total

This test is worth 20% of your final grade. This test is open book and open notes. Use the back of the sheet if you need more room to write, although you shouldn’t need to. State your assumptions wherever it makes a difference. You have 60 minutes.

I. True False: (2 pts. each)

T F 1. A class is abstract only if all its methods are abstract.
T F 2. When implementing an interface you only have to give definitions for those methods declared as abstract.
T F 3. Java parameters are by their very nature reference parameters.
T F 4. A single try block can have multiple catch blocks.
T F 5. Exceptions can occur as a side-effect to some computation in a program or they can be thrown explicitly.
T F 6. Exceptions can be handled locally or thrown.
T F 7. To tell whether an exception is checked or unchecked you can simply look at its declaration.
T F 8. A JFrame gives basic window functionality such as a window title and buttons to maximize or close a window.
T F 9. Interfaces can be used as a mechanism to share constants between classes.
T F 10. An event handler can respond to events from multiple objects.
T F 11. Multiple event handlers can be registered for a single event from a single object.
T F 12. Class A could create an instance of an inner class declared in class B.
T F 13. An Adapter class is commonly used to avoid having to implement all 7 methods from the MouseListener and MouseMotionListener classes.
T F 14. In non-Swing Java graphics, overriding the update() method is recommended to prevent the window from flickering due to background refreshes.
T F 15. A netbean is a snippet of code designed to give access and funcionality to a Java class instance variable for use on the web.
T F 16. Animation can be achieved by moving a still image across the window.
T F 17. Method paint() must be overridden in an applet for it to function properly.
II. Problems: (6 pts. each) Briefly answer each of the following.

1. Consider the following loop in ALGOL 60, as discussed in class.

   \[ \begin{align*}
   i &:= 1; \\
   &\text{for } \text{count} := 1 \text{ step } \text{count} \text{ until } 3 * i \text{ do} \\
   &\quad i := i + 1
   \end{align*} \]

   Give the equivalent code in a single for loop in Java:

   \[ \begin{align*}
   \text{int } &i = 1; \\
   \text{for } &\left( \text{int } \text{count} = 1; \text{count} < (3 * i); \text{count} += \text{count} \right) \\
   &\quad i = i + 1
   \end{align*} \]

2. Consider the following code written using C syntax, though not C parameter semantics:

   ```c
   int value; /* Global variable */
   int list[4]; /* Global array */

   void swap (int a, int b)
   {
     int temp = a; list[a] = b;
     list[a] = temp;
   }

   void main()
   {
     value = 3; list[0]=1; list[1]=0; list[2]=3; list[3]=2;
     swap( list[0], list[1] );
     swap( value, list[value] );
   }
   ```

   What are the values of the variable "value" and the array "list" at the end of the above program if parameters are passed by name?:

   \[
   \begin{array}{cccccc}
   \text{value} & \text{list[0]} & \text{list[1]} & \text{list[2]} & \text{list[3]} \\
   \hline
   \text{2} & \times & 0 & 3 & 2 \\
   \end{array}
   \]
3. Consider a class called SampleClass. What would you not be able to do if the toString() method is missing?

Display meaningful information using System.out.println(...) A non-meaningful number from a subclass toString() method is given as default.

4. In other programming languages such as C, we can handle errors using a return value or by printing an error message and exiting the program. Name 3 advantages that Java exception handling has over error message handling in languages such as C

a) Separating error handling from "regular" code

b) Propagating errors up the call stack

c) Grouping error types and error differentiation.

5. What is the difference between putting code in the exception-handling finally block and simply listing the code after the try-catch block?

If there is an exception, the code in the finally block will still be executed. Code simply listed after the try-catch block would not be executed after an exception.

6. For error-checking using a computationally expensive section of code which is better: assertions or writing your own error-handling code? You must explain your answer for credit.

assertions are better, since we can disable them at compile time.
7. Assume you overhear a CS 340 student saying "I don’t understand what the big deal is with double-buff-
ering. In both cases we are writing to memory, so neither one should one be faster than the other." Do
you agree or disagree? Briefly explain your reasons.

Disagree: Writing to memory is faster than updating video memory. Double-buffering accumulates the changes, and then reinitializes the window all at once.

8. Do the same issues with double-buffering exist with AWT classes and Swing classes? Explain.

With AWT classes you must explicitly implement double-buffering. In comparison, double-buffering is built-in to Swing classes.

9. Assume you have a window with three buttons in it layed out something like this:

```
A B C
```

a) Draw a picture of what this interface would look like if you are using a FlowLayout Manager and the window is resized to be narrow and tall.

```
A
B
C
```

b) Draw a picture of what this interface would look like if you are using a BorderLayout Manager and the window is resized to be narrow and tall.

```
A  C
   B
```
10. One of the nice features of the Collections classes is that we can take an existing data structure and use it in the constructor for another type from the Collections class. Assume that an ArrayList has been created (e.g. ArrayList myArrayList) and initialized with some values, where some of them are duplicates.

Assume that a HashSet is created using the ArrayList that already exists:

```java
Set myHashSet = new HashSet(myArrayList);
```

One of the characteristics of class Set is that there are no duplicate elements. When looking at the elements stored in myHashSet though, you notice two problems: 1. There are duplicates in the list, and 2. The order of the elements doesn’t seem quite right.

a) What is the most plausible explanation for there being duplicates?

```
equals() was not implemented.
```

b) What is the most plausible explanation for the order of elements not being quite right?

```
hashCode() was not implemented.
```

c) Both HashSet and TreeSet are subclasses of Set. TreeSet, however, is ordered. Once HashSet (described above) is working properly, assume that we create a new TreeSet as:

```java
Set myTreeSet = new TreeSet(myHashSet);
```

When looking at the elements they are not ordered! What is the most plausible explanation as to why they are not ordered?

```
the Comparable interface (implementing compareTo()) is not implemented.
```