CS365 Midterm

1) You must answer *all* of the questions.
2) Place your name on the exam.
3) You must use Java to implement the coding questions.
4) Good luck!

1. **(22 points)** Choose the appropriate term from the following list to fill in each blank below:

abstract type	abstract class	class	composition	concrete type	Error	Exception
executable	extends	friend	generic	hash tables	implementation	implements
inheritance	interface	jar	java	jump tables	library	longjmp
module	namespace	non-virtual	Object	object-oriented	overloading	package
parametric	private	protected	public	pure virtual	replicated	Runtime Exception
setjmp	shared	source	static	subclass	subtype	superclass
supertype	tar	template	Throwable	throws	types	virtual
virtual machine	void	vtables	…			

a. **parametric** polymorphism uses templates instantiated with types while **subtype** polymorphism makes use of class hierarchies to support polymorphism.

b. **Replicated** inheritance refers to multiple inheritance in which a subclass receives multiple copies of any shared superclasses.

c. C++ attempts to provide some of the benefits of packages/modules through its **namespace** and **friend** mechanisms.

d. A **virtual machine** is an interpreter that simulates the execution of a machine by executing the machine's instruction set using software piece of software that simulates machine instructions

e. **Exception** If you create a user-defined exception class in Java, which Java class should it subclass?

f. In Java, an interface is used to declare a(n) **abstract type**.

g. **vtables** are used to implement virtual methods at run-time

h. **longjmp** The C function that takes stored state information and an error code as arguments, and that throws control back to the calling function by popping the stack and restoring the state information of the calling function.

i. A **jar** file in Java allows a collection of classes to be bundled together and treated like a single file for execution, much like a binary file in C/C++.

2. **(3 points)** Suppose that class Goo wants to restrict access to its instance variable size to 1) any class in its package and 2) any subclass, whether that subclass is in Goo's package or another package. What access protection should the programmer assign to size?
   a. private
   b. protected
   c. package
   d. public

3. (**4 points)** Suppose you have the following C++ classes:
   Class Fruit { … }
   Class Melon : public Fruit { … }
   Class Cantelope : public Fruit { … }

   Further assume that Fruit is an abstract class because it has one or more pure virtual methods. Circle **all** all of the following declarations that are legal:
   a. Fruit f;
   b. Fruit f = Fruit();
   c. Fruit f = Melon();
   d. Fruit *f = new Fruit();
   e. Fruit *f = new Melon();
   f. Fruit *f = new Cantalope();

4. **(20 points--Generics)** Write a Java template class named Stack that takes a single type parameter named E and supports three methods:
   a. a constructor that initializes the stack.
   b. pop: takes no parameters, removes the first value from the top of the stack and returns it. If the stack is empty it should throw an instance of a class named StackEmpty back to the calling function. You do not need to know the particulars of StackEmpty and you should assume that that class already exists.
   c. push: a void method that takes a single parameter of type E and assigns it to the top of the stack.

```
class Stack<E> {
  class StackNode<E> {
    E value;
    StackNode<E> next;
  }
  StackNode<E> top;

  Stack() {
    top = null;
  }

  E pop() throws StackEmpty {
    if (top == null)
      throw new StackEmpty();
    E returnValue = top.value;
    top = top.next;
    return returnValue;
  }

  void push(E item) {
    StackNode<E> newNode = new StackNode<E>();
    newNode.value = item;
    newNode.next = top;
    top = newNode;
  }
}
```

5. **(21 points--Inheritance)** Suppose that you are designing the input portion of a Java application. The application should be able to read from the 1) console, 2) a file, or 3) a widget such as a type-in text box. You have been given the following specifications:
   - Regardless of which input object you are using, the application needs to be able to open/close the object (open and close methods that take no parameters and return void).
   - The application needs to be able to read a text string from an input object (read method that returns a string and takes no parameters).
   - The classes for the reader objects should be ConsoleReader, FileReader, and TextReader.
   - The open and close methods are the same for each of the three classes but the read method is different for each class.

   a. Declare an interface named Reader so that objects of any of the three classes can be assigned to a Reader variable. For example:

      Reader myReader = new ConsoleReader();

      interface Reader {
      void open();
      void close();
      String read();
      }

   b. Design and draw a class hierarchy for the above three objects using Java.
      - Next to each class list the methods that you would declare with that class. You should list a method with a class only if you would provide an implementation for that method in that class.
      - Put an asterisk next to any method that should be declared abstract

```
                        AbstractReader
                             open
                             close
                             read*
                               |
       -------------------------------------------
       |                       |                 |
 ConsoleReader             FileReader        TextReader
    read                      read              read
```

Since the implementation for open and close is the same for all three Reader objects, you should create a superclass that declares and provides an implementation for these two methods. The read method does not have to be declared in AbstractReader, but in that case you must ensure that

ConsoleReader, FileReader, and TextReader all implement the Reader interface.

c. Provide Java class declarations for any superclasses that you created in (b) and for ConsoleReader.
    a. The classes must implement the Reader interface
    b. You should *not* show any implementation. For example, to declare a void method named foo, write:

```
        public void foo();
```

```
abstract class AbstractReader implements Reader {
    public void open() { ... }
    public void close() { ... }
    abstract public String read();
}

class ConsoleReader extends AbstractReader {
    public String read() { ... }
}
```

or if you choose not to declare the read method in AbstractReader:

```
class AbstractReader {
    public void open() { ... }
    public void close() { ... }
}

class ConsoleReader extends AbstractReader implements Reader
{
    public String read() { ... }
}
```

Note that the latter solution is not as good because you can actually create an instance of AbstractReader, which is undesirable. You cannot fix the problem by declaring AbstractReader to be abstract, since the Java compiler will complain.

6. **(25 points--Java Programming)** Write a complete Java program, including import statements in a class called Grader that meets the following specifications:
   a. The class should be part of the package **Grade**.
   b. Use the constructor to implement whatever you would normally implement in main in a C++ program. The constructor should take a single string argument which is the name of the grade file.
   c. Your program should read scores and student names from a grade file whose name has been provided as a command line argument and for each student it should print the student's name and average score to stdout. The average should be computed as an integer average. For example, if the student's scores are 7 and 10, then the printed average should be 8.
   d. Your constructor should not handle the IOException that may be generated by opening the file but your main method should catch the exception and print the exception's message by invoking its getMessage() method.

Each line of the grade file lists a student's name and then the student's scores. The grade file may have multiple students. For example:

```
Baby Daisy 59 75 93 53
Smiley The Amazing Hound 86 45 100 63 78 91
Chipmunk 45
```

Note that a name may consist of an arbitrary number of words and that a student may have an arbitrary number of scores.

Your output should be formatted as follows:

1. The name should be left-justified in a field 30 characters wide.
2. The average should be right-justified in a field 3 characters wide.
3. There should be a space between the two fields.

For the above input, your program would produce the output:

```
Baby Daisy                      70
Smiley The Amazing Hound        77
Chipmunk                        45
```

Here is some additional information:

1. You are guaranteed that:
   - there is at least one score for each student,
   - that all scores are non-negative integers, and
   - that every line starts with at least one word.
2. The API for the Scanner class has been provided at the end of this exam.
3. You should use the FileReader class for a file. You do not need the API for the FileReader class in order to complete this problem.
4. The Scanner class is in Java's util library and the FileReader class is in Jave's io library

```
package Grade;

import java.util.Scanner;
import java.io.*;

class Grader {
  Grader(String filename) throws IOException {
    Scanner reader = new Scanner(new FileReader(filename));
    Scanner tokenizer;
    String line, name;
    int sum, count;
    while (reader.hasNextLine()) {
      line = reader.nextLine();
      tokenizer = new Scanner(line);
      name = tokenizer.next();
      while (!tokenizer.hasNextInt()) {
        name = name + " " + tokenizer.next();
      }
      sum = 0;
      count = 0;
      while (tokenizer.hasNextInt()) {
          sum += tokenizer.nextInt();
          count++;
      }
       tokenizer.close();
       System.out.printf("%-30s %3d%n", name, sum / count);
    }
  }
  static public void main(String args[]) {
    try {
      new Grader(args[0]);
    }
    catch (IOException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

7. **(5 points: Executing a Java Program)** Answer the following questions about the previous problem.

   a) What is the name of the file in which Grader should be placed?
   Grader.java

   b) What is the name of the directory in which Grader's file should be placed?
   Grade

   c) Suppose that I am in an arbitrary directory and that the directory containing Grader's file is stored in a directory named /home/bvz/labs. Further assume that the grade file is named input.txt. Write the full java command required to execute the Grader program that you wrote in the previous question.

   Any of the following answers are acceptable:

   java –cp .:../home/bvz/labs Grade.Grader input.txt
   java –cp /home/bvz/labs Grade.Grader input.txt

   java –classpath .:../home/bvz/labs Grade.Grader input.txt
   java –classpath /home/bvz/labs Grade.Grader input.txt

   -cp is an abbreviation for –classpath. You also don't have to include the .:.. prefix since you know that the Grade package is in /home/bvz/labs.