CSE 332 Studio Session on Advanced Topics on Inheritance

These studio exercises are intended to introduce advanced C++ inheritance features: multiple inheritance, and virtual base classes; and to give you experience using those concepts and techniques within the Visual C++ environment. In this studio you will again work in self-selected small groups of 2 or 3 people. As before, students who are more familiar with the material are encouraged to help those for whom it is less familiar. Asking questions of your professor and teaching assistants (as well as of each other) during the studio sessions is highly encouraged as well. Please record your answers you work through the following exercises. After you have finished please send your answers to the required exercises, and to any of the enrichment exercises you completed, in an e-mail to the cse332@seas.wustl.edu course e-mail account, with the subject line "Advanced Inheritance". The enrichment exercises are optional but are a good way to dig into the material a little deeper, especially if you breeze through the required ones.

PART I: REQUIRED EXERCISES

1. Find/choose your team members in the studio area, log in to one of the Windows machines, open up Visual Studio, and create a new Visual C++ Win32 Console Application project for this studio. Change the signature of the main function in the source file that Visual C++ generated to match the one that was used in the lecture slides on C++ program structure, which are available on the course web page. Write down the names of the team members who are present (if a team member arrives late please catch them up on the work, and add their name) as the answer to this exercise.

2. Declare and define an **`Animal`** class and an **`Endangered`** class, each with a constructor, a virtual destructor, and a pure virtual **`print`** method that takes no parameters and has a **`void`** return type. Define the **`print`** method of the **`Endangered`** class so that it prints out "**`ENDANGERED!`**" when it is called. The body of each constructor and destructor of each class should contain a statement that prints the class and method name (e.g., the **`Animal`** destructor should print out something like "**`Animal::~Animal`**"). Also declare and define a **`Bear`** class that is derived from **`Animal`** via public inheritance and has a constructor and a virtual destructor, each of which should contain a statement that prints the class and method name, and a pure virtual **`toes`** method that takes no arguments and returns an **`int`** by value. In your main function try to declare an object of each of those classes (and for any that you cannot instantiate commenting out the declaration of the object of that type), and as the answer to this exercise please explain for each of those classes (1) what happens when you do that and (2) why you think that happens.

3. Declare and define **Panda** and **Grizzly** classes that are derived via public inheritance from both **Bear** and **Endangered**. The **Panda** class should have a **string** member variable for the animal's name (which the constructor should initialize with the value of a parameter it takes) and each class should have a constructor and a destructor whose bodies contain a statement that prints out the class and method names to the standard output stream. Each class should override the base class **toes** and **print** methods: the **toes** method should return the value 6 (a number including all 5 digits and the famous "panda's thumb") for a **Panda** and 5 for a **Grizzly;** each **print** method should first call the **Endangered** class **print** method and then output the type of animal it is (and for a **Panda** also the animal's name). In your main function, declare an object of type **Panda** initialized with a string of your choice for the animal's name and an object of type **Grizzly** that is default initialized. Build and run your program, and as the answer to this exercise please show your program's output.

4. Add a non-virtual **cuddle** method to your **Panda** class, and a non-virtual **growl** method to your **Grizzly** class. Define each of those methods simply to print out the class and method names. In your main function try to call each of the following methods on both the **Panda** and **Grizzly** objects: **toes**, **print**, **growl**, and **cuddle**. Comment out any of the calls that cannot be made on a particular object. As the answer to this exercise please say which calls could be made on each object and for those that could be made please show the output produced by each call.

5. Add a pure-virtual **react** method to the **Animal** class: in your **Panda** class override it to call the **cuddle** method, and in the **Grizzly** class to call the **growl** method. In your main function call the **react** method on each of the **Panda** and **Grizzly** objects, and as the answer to this exercise please show the output produced by each call.

6. Repeat exercise 5, but instead of calling the **react** method directly on each object, store the addresses of the **Panda** and **Grizzly** objects in separate variables of type **Animal *** and then make the call to that method using each pointer. Then try to call the **growl** and **cuddle** methods using each pointer, again commenting out any calls that do not work. As the answer to this exercise please explain what remained the same and what changed, compared to the results you observed in exercises 4 and 5.

PART II: ENRICHMENT EXERCISES (optional, do the ones that interest you).

7. Repeat the previous exercise, but instead of declaring the **Panda** and **Grizzly** objects on the stack, allocate them on the heap and store their addresses in separate shared pointers to **Animal**. Try the same calls as in the previous exercise, though the shared pointers instead of through built-in pointers. As the answer to this exercise please describe what if anything changed when you did that.

8. Repeat the previous exercise, with all destructors non-virtual. As the answer to this exercises please explain (1) how your program's output changed when you did that, and (2) why you think that happened.