Workshop 8 reflections

https://isocpp.org/wiki/faq/abcs

1. What is the difference between a abstract class and concrete class?

An abstract class contains one or more pure virtual functions (functions with a declaration, but no body). And what is a pure virtual function (or PVF)? A PVF is a function with a head and no body. e.g.

```
class Account {
// other stuff....
virtual void foo() = 0;       // a pure virtual function. Account::foo() cannot have a body.
///....
```

The =0 makes it a pure virtual function.

Now you can compile your code without the compiler warning you that “missing definition for void foo()”. You make a class concrete by implementing ALL of it’s missing pure virtual functions. A concrete class contains NO pure virtual functions, not even one. You do this by adding a function declaration in the derived class

```
class Checking {
// other stuff...
virtual void foo()   { cout << “Hello from foo” ; }   // a virtual function with a body
```

In this lab, since Account is an Abstract Base Class (ABC) there can never be an Account object. But there can be an Account pointer to a derived object (of type Checking or Saving). The only functions in those classes that you access through an Account * are the virtual functions declared inside the Account class.

```
Account a;          // error! Cannot instantiate an ABC.
Account* p;         // ok, make it point to a Checking or Saving object.
Checking C(0);
p = &C;             // ok, now we can call functions in C through the Account pointer *p.
```

2. Why do we need a PVC in a base class?

1. Well all the derived classes in this workshop share the same base class, so if you want to force all of them to share the same declaration for a function you HAVE to declare it in the base class. The base class becomes like an interface that they all are forced to follow.

2. for consistency. Abstract classes are meant to be extended. The same function declaration in the base class forces the child classes to use the same declaration, so they all do it the same way.

A pure virtual function allows us to separate the definition of how we interact with a class from the implementation of that class.

3. Explain what have you learned in this workshop.

How to code generic behaviour in a base class, and specific behaviour in derived classes.

How to make an interface. main() only needs to use the functions defined in **Account**, but main() works for any type of class derived from **Account**, such as **SavingsAccount** and **CheckingAccount**.