Handout 5  
Introduction to Programming in C++

Deadline is November 29

**Exercise 1:** Given the following C++ code:

class A {  
public:  
 int x;  
 A \*objARef;  
private: int y;  
protected: int z;  
}; ------------------------------------  
class B: public A {  
public: A objA;  
}; ------------------------------------  
class C {  
public:  
 A objA;  
 A \*objARef;  
 B objB;  
};

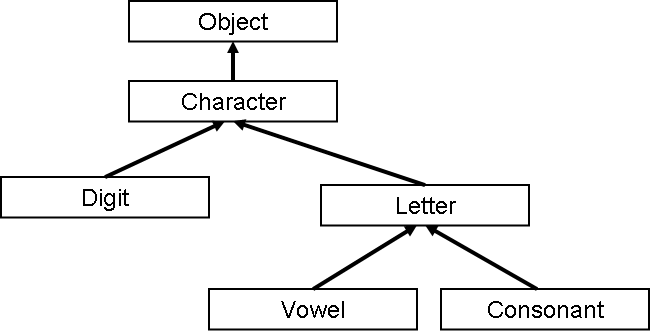
Determine for each of the attribute-access-expressions whether it results in an Error.

* A variable in the superclass that is private is not visible in the subclass
* A variable in the superclass that is protected is visible in the subclass
* All variables are visible of an object that is part of its class
* A variable of an object that is part of its classes’ superclass / or its superclass in the superclass that is protected is not visible in the subclass

|  |  |  |  |
| --- | --- | --- | --- |
|  | **in class A** | **in class B** | **in class C** |
| **x** | OK | OK | Wrong |
| **y** | OK | Wrong | Wrong |
| **z** | OK | OK | Wrong |
| **objA.x** | Wrong | OK | OK |
| **objA.y** | Wrong | Wrong | Wrong |
| **objA.z** | Wrong | Wrong | Wrong |
| **objARef->x** | OK | OK | OK |
| **objARef->y** | OK | Wrong | Wrong |
| **objARef->z** | OK | Wrong | Wrong |
| **objB.x** | Wrong | Wrong | OK |
| **objB.y** | Wrong | Wrong | Wrong |
| **objB.z** | Wrong | Wrong | Wrong |

**Exercise 2:**

Given the following class hierarchy:



1. Create C++ code without attributes and methods for all for all 6 classes.

2. Extend the class character by a public attribute **ch**, so that it can store a single character.

3. Overload the operator + for the class Character, so that it can add two objects of type Character. Implement the + by summing up the numeric ASCII-codes of two characters followed by the application of a ‘modulo 128’. Similar to:

**char sum(char c1, char c2) { return (char)(((int)c1 + int(c2))%128); }**

4. Override the operator + in the Digit class, so that it adds the numeric value of two digits and delivers the digit that we get if we finally apply “modulo 10”.

(Example ‘5’ + ‘6’ = ‘1’ // 5 + 6 = 11 % 10 = 1)

5. Extend the Object class by an object counter that counts the number of created objects for all objects of the above class hierarchy. (Tip: Lecture 9 slide 5) The counter should be embedded into the Object-class default constructor.

6. Change the visibility of the attribute **ch**, so that it is visible in all subclasses, but inaccessible from outside. Create a get-set method pair for the attribute **ch**.

7. Create a main-method, where you create 2 objects of each class in the above class hierarchy and that prints finally the value of your object counter (**this should be 12**).

#include <iostream>  
  
class Object {  
public:  
 static int counter;  
public:  
 Object() {  
 counter++;  
 }  
};  
  
class Character : public Object {  
protected:  
 char ch; // 2.  
public:  
 Character operator+(Character &c) {  
 Character result;  
 result.ch = (char) (((int) this->ch + ((int) c.ch)) % 128);  
 return result;  
 }  
  
 char getCh() {  
 return this->ch;  
 }  
  
 void setCh(char ch) {  
 this->ch = ch;  
 }  
};  
  
class Digit : public Character {  
public:  
 Digit operator+(Digit &d) {  
 Digit result;  
 result.ch = ((((int) (this->ch - 48)) + ((int) (d.ch - 48))) % 10) + 48;  
 return result;  
 }  
};  
  
class Letter : public Character {  
  
};  
  
class Vowel : public Letter {  
  
};  
  
class Consonant : public Letter {  
  
};  
  
int Object::counter = 0;  
  
int main() {  
 Object object1;  
 Object object2;  
  
 Letter letter1;  
 Letter letter2;  
  
 Vowel vowel1;  
 Vowel vowel2;  
  
 Consonant consonant1;  
 Consonant consonant2;  
  
 Character a;  
 a.setCh('a');  
 Character b;  
 b.setCh('b');  
 //std::cout << "Character: " << (a + b).getCh() << std::endl; // 3.  
  
 Digit one;  
 one.setCh('1');  
 Digit two;  
 two.setCh('2');  
 //std::cout << "Digit: " << (one + two).getCh() << std::endl; // 4.  
  
 std::cout << "Counter: " << Object::counter << std::endl;  
  
 return 0;  
}