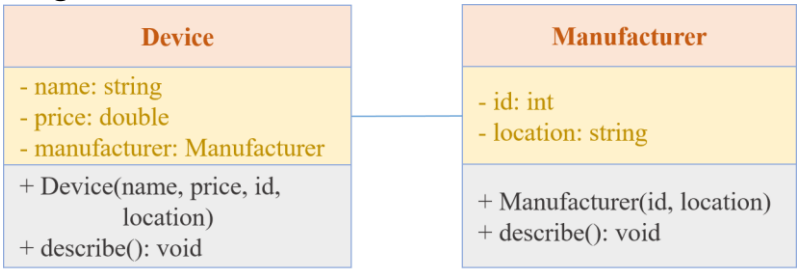
VIETNAMESE-GERMAN UNIVSREITY
COMPUTER SCIENCE PROGRAM

Final Test
Course: Object Oriented Programming - C++
Duration: 120 minutes

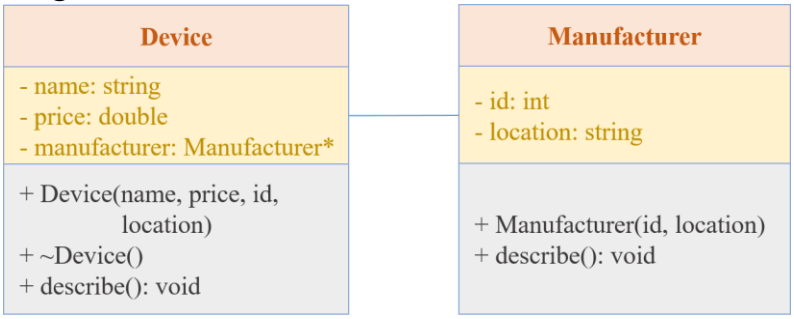
Submit your code (.cpp files) to the classroom before 4:00 PM
You could use GG Search or eBooks.
Do not open any kinds of media, such as Facebook, Skype, etc., during the exam.

1) Implement the following class diagrams. Please note that you need to implement exactly what is described in the class diagrams. You cannot create a constructor with a difference number of arguments or something like that.

a) Class diagram 1



b) Class diagram 2



Test sample for both (a) and (b)

```
int main() {
    // create a mouse with
    // name="mouse" and price=2.5
    // id=9725 and localtion="Vietnam");
    mouse.describe();

    return 0;
}
```

```
/tmp/RUc70aJTcy.o
Name: mouse - Price: 2.5
ID: 9725 - Location: Vietnam
```

2) Write a program to create a class called Beverage which is used to represent prices of a beverage. The class will have three private float data members (price, topping price, and condiment price). Add the following member methods:

a) a default constructor (set value of data members to zero);

b) a constructor with parameters;

c) a copy constructor;

d) overloaded plus operator (+), it must add price to price, topping price to topping price, and condiment price to condiment price;

e) overloaded comparison operator (==), two beverages are considered to be equal, if the three values of data members are equal;

Note: You need to implement the describe() method to print out the content of the objects (print values of price, topping price, and condiment price).


```cpp
int main() {
    // create Beverage b0 with default constructor
    b0.describe();

    // create Beverage b1 with
    // price=10.5 topping_price=2.3
    // condiment_price=1.2
    b1.describe();

    // create Beverage b2 with
    // price=15 topping_price=2.2
    // condiment_price=0.5
    b2.describe();

    // create Beverage b3 by using copy contructor
    // copy b1 into b3
    b3.describe();

    // overloaded plus operator (+)
    // b3 = b1 + b2
    b3.describe();

    // overloaded comparison operator (==)
    // Is b3 == b1?
    cout << "Is b3 = b1? " << (b3 == b1) << endl;
    // assign b1 to b3
    // Is b3 == b1?
    b3 = b1;
    b3.describe();
    cout << "Is b3 = b1? " << (b3 == b1) << endl;
    return 0;
}
```

```
price: 0 topping_price: 0 condiment_price: 0
price: 10.5 topping_price: 2.3 condiment_price: 1.2
price: 15 topping_price: 2.2 condiment_price: 0.5
price: 10.5 topping_price: 2.3 condiment_price: 1.2
price: 25.5 topping_price: 4.5 condiment_price: 1.7
Is b3 = b1? 0
price: 10.5 topping_price: 2.3 condiment_price: 1.2
Is b3 = b1? 1
```

3) A Ward consists of a name (string) and a list of people. A person in this system can be a student, a doctor, or a teacher. A student comprises a name, year of birth (int), and a grade (string). A teacher consists of a name, year of birth, and a subject (string). A doctor includes a name, year of birth, and a specialist (string). You need to use a **vector** to store a list of people.

a) Implement classes for the above description. Implement the describe() method to print out the content of the objects for all classes.

b) Write the **addPerson(Person* person)** method in the Ward class to add a new person to the ward (list of people). Create a ward object, then adding one student, two teachers, and two doctors. Implement the describe() method to print out the ward name and the information of people in the ward.

c) Write the countDoctor() method to **count** a number of **doctors** in the ward.

d) Write the sortAge() method to sort people in the ward by their **age** in increasing order. Please remember to include <algorithm>.

e) Write the aveTeacherYearOfBirth() method to calculate the **average year of birth** of **teachers**.

Test sample is on the next page

```cpp
int main () {
   // create a student with
   // name="studentA", yearOfBirth=2010, grade="7"
   student.describe();
   // create teacher1 with
   // name="teacherA", yearOfBirth=1969, subject="Math"
   teacher1.describe();
   // create doctor1 with
   // name="doctorA", yearOfBirth=1945, specialist="Endocrinologists"
   doctor1.describe();

   // create teacher2 with
   // name="teacherB", yearOfBirth=1995, subject="History"
   // create doctor2 with
   // name="doctorB", yearOfBirth=1975, specialist="Cardiologists"

   // create ward with name="Ward1"
   // add 1 student, 2 teachers, 2 doctors
   ward.addPerson(&student);
   ward.addPerson(&teacher1);
   ward.addPerson(&teacher2);
   ward.addPerson(&doctor1);
   ward.addPerson(&doctor2);
   ward.describe();

   // use countDoctor() method to count
   // number of people and store the result
   // in numberOfDoctors variable
   cout << "number of doctors: " << numberOfDoctors << endl;

   // sort people in the ward by their age
   // use describe method to show the sorted result
   ward.describe();

   // use aveTeacherYearOfBirth() method to count
   // number of people and store the result
   // in aveTeacherYearOfBirth variable
   cout << "average year of birth (teachers): " << aveTeacherYearOfBirth << endl;

   return 0;
}
```

```
Name: studentA - yearOfBirth: 2010 - Grade: 7
Name: teacherA - yearOfBirth: 1969 - Subject: Math
Name: doctorA - yearOfBirth: 1945 - Specialist: Endocrinologists
Ward Name: Ward1
Name: studentA - yearOfBirth: 2010 - Grade: 7
Name: teacherA - yearOfBirth: 1969 - Subject: Math
Name: teacherB - yearOfBirth: 1995 - Subject: History
Name: doctorA - yearOfBirth: 1945 - Specialist: Endocrinologists
Name: doctorB - yearOfBirth: 1975 - Specialist: Cardiologists
number of doctors: 2
Ward Name: Ward1
Name: studentA - yearOfBirth: 2010 - Grade: 7
Name: teacherB - yearOfBirth: 1995 - Subject: History
Name: doctorB - yearOfBirth: 1975 - Specialist: Cardiologists
Name: teacherA - yearOfBirth: 1969 - Subject: Math
Name: doctorA - yearOfBirth: 1945 - Specialist: Endocrinologists
average year of birth (teachers): 1982
```