Duration: 2 hours .

Name: Date: .
Allowance: Open Book Exam.

Object Oriented Programming - Sample Exam

Q.1 True or False:

1. If the class **`Y`** is a sub-class of the class **`X`**, then an object of the class **`Y`** will only inherit the protected members declared in the class **`X`** . F
2. A **`public`** member of a class can not be directly accessed by its name in the main() function. F
3. A friend function is allowed to access the private members of a class, but not the protected members of the class. F
4. If a programmer does not declare a constructor for a certain class, then the compiler will automatically create a default constructor for that class. T
5. In order to access a static variable of a class we have to write the class name before the variable name. T
6. The ***this*** pointer is a pointer that points to the object that the method has been called for. T
7. Function overloading is considered as one of the ways of polymorphism in object oriented programming. T
8. The constructor of a class is a member function that returns void. F
9. Encapsulation means extending a class to increase more member functions and member variables in a derived class. F
10. When a certain value is assigned to a static member variable at some point of the program, then it is not possible to assign a different value to this variable at a further point in the program. F
11. When we overload a function, we declare another function with the same name and we must specify a different number of parameters. T

Q.2 Choose the correct answer:

1. **A constructor is:**
   a) A member function that removes an object from the memory.
   b) A member function that is called automatically when an object is being removed from the memory.
   c) A member function that is used to initialize variables of a declared object.
   d) None of the above.

2. **Given the class below, what changes do we need to make to the class in order to have a facility to test if the values of two objects' data are equal?**

```
class MyClass
{
   private:
           float a ;
```

```
            float b ;
   public:
         MyClass( )

               a = 0 ;
               b = 0 ;
         }

         void setValues(int x , int y)
         {
               a = x ;
               b = y ;
         }
};
```

a) Declare a third temporary member integer.
b) Do operator = = overloading.
c) Create a new member function: **`void testEqual( )`**, that takes no parameters.
d) None of the above.

3. **Which of the following statements are true about destructors?**
a) Destructors are special functions with the same name of the class and are preceded by a tilde character (~).
b) Destructors cannot be overloaded.
c) A destructor has no return type.
d) All of the above.
e) None of the above.

4. **Which of the following most closely describes the process of overloading?**
a) A class with the same name replaces the functionality of a class defined earlier in the hierarchy.
b) A method with the same name completely replaces the functionality of a method defined earlier in the hierarchy.
c) A method with the same name but different parameters gives multiple uses for the same method name.
d) A class is prevented from accessing methods in its immediate ancestor.

5. **What will happen when you attempt to compile the code below?**

```
class MyClass
{
         int x ;

   public:
         int y ;
};

void main()
{
   MyClass obj ;
   obj.x = 100 ;
```

```
    obj.y = 500 ;
}
```

a) Compiler Error because x is not accessible.
b) Compiler Error because y is not accessible.
c) Compiler Error because **MyClass** is an abstract class and can not be instantiated.
d) a and **b**.
e) The code will compile successfully.

6. **'A plane is a machine that has a motor and has wings'.**
   'A refrigerator is a machine that has a motor and has shelves'.
   Which of the following best describes the previous statements as a set of classes?
   a) 3 classes: A **machine** class that has one attribute: *motor*. A **plane** class that inherits from the **machine** class. And a **refrigerator** class that inherits from the **plane** class.
   b) 3 classes: A machine class that has one attribute: motor. A plane class that inherits from the machine class. And a refrigerator class that also inherits from the machine class.
   c) 2 classes: A plane class that has two attributes, and a refrigerator class that also has two attributes.
   d) 1 class: A **machine** class that has an attribute for the *type of machine*.

7. **The following function prototype uses the default arguments feature:**
   `void myFunc(int x=3 , int y) ;`
   is the function valid this way?
   a) Yes, it is ok to give a default parameter for **x** and not for **y**.
   b) No, a function must always declare all its variables with default values.
   c) No, you must only give a default parameter for **y** but not for **x**.
   d) No, there should not be any arguments without default values to the right of the default arguments.

8. **A derived class can directly access:**
   a) The protected and public members of the base class.
   b) The private and protected members of the base class.
   c) Only the protected members of the base class.
   d) Only the private members of the base class.
   e) The public, private, and protected members of the base class.

9. **Assume you have a class `X` that contains an object of class `Y`. Assume that we declare an object of class `X` in the main( ) function. When will the body of the constructor of class `Y` be executed?**
   a) When any member function of the class **`X`** is called.
   b) After the body of the constructor of class **`X`** is executed.
   c) Before the body of the constructor of class X is executed.
   d) We can not determine exactly when it will be executed.

10. **An embedded (aggregated) object is:**
    a) An object that s declared with the default constructor.
    b) An object that is included by another object.

c) An object from a class that is inherited from another class.
d) b and **c**.
e) None of the above.

Q.3 Answer the following question:

Using your own words, explain the following in brief:
(i.e. discuss in a few lines or a short paragraph…. You don't need to write too much!):

A) What is the difference between **Multi-Level Inheritance** and **Multiple Inheritance**?

B) What are the core features of Object Oriented Programming?

C) What is the meaning of aggregation relation? What is the effect of such relation on the sequence of execution of the constructors?

D) What are the characteristics of a constructor?

Q.4 Answer the following question:

Design the set of classes that would best represent the following situation:

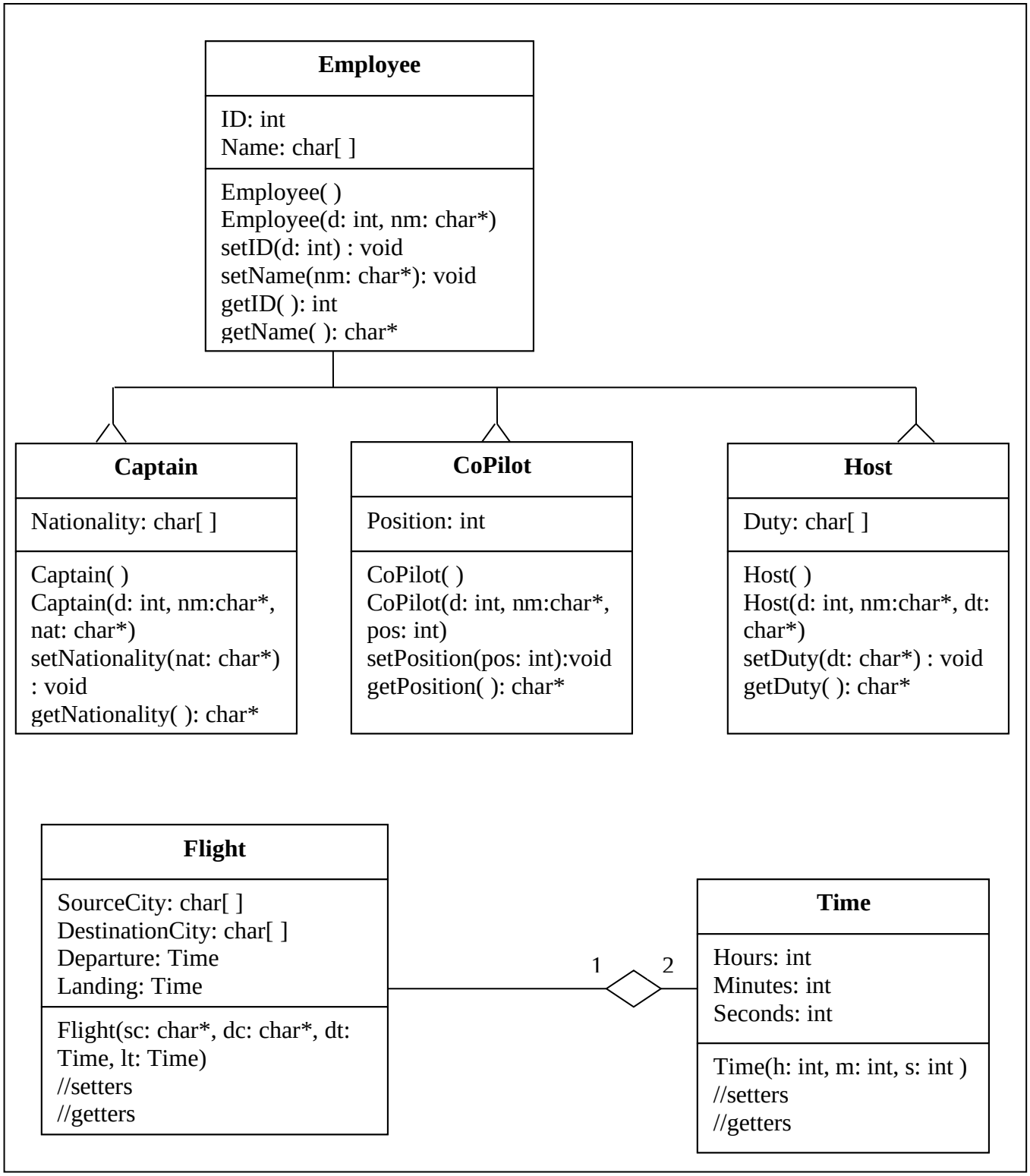
(Note: ***Design*** *means*: you can either draw a **UML diagram**. **OR** write the **class body**.
Whichever way you choose, make sure that you show the **data types** of the **attributes**, and write the **prototypes** of the **constructors** and **methods**. **YOU DO NOT NEED** to write any code inside the body of the methods).

You are asked to design a set of classes for an Airlines Company information system. The company has many members of staff: Captains, Co-Pilots, and Hosts. The company also wants to store information about its flight journeys. The required information is shown in details below:

Captain: ID, Name, Nationality .
Co-Pilot: ID, Name, Position.
Host: ID, Name, DutyDetails.
Flight: SourceCity, DestinationCity, Time of departure, Time of landing.

Hint: You can make any additional assumptions when needed. If you do so, then write one sentence about your assumption.

Solution using a simple UML diagram: (either use UML or write the class, but not BOTH!)



Assumptions: We will assume that Time is a new datatype.

Other solution by writing the class with the function prototypes:

```
Class Employee
{
   private:
        int ID ;
        char name[50] ;
   public:
        Employee( ) ;
        Employee(int, char*) ;
        void setID(int) ;
        void setName(char*) ;
        int getID( ) ;
        char* getName( );
}

class Captain : public Employee
{
   private:
        char nationality[20] ;
   public:
        Captain( ) ;
        Captain(int, char*, char*) ;
        void setNationality(char*) ;
        char* getNationality( ) ;
}

class CoPilot : public Employee
{
   private:
        int position ;
   public:
        CoPilot( ) ;
        CoPilot(int, char*, int) ;
        void setPosition(int) ;
        int getPosition( ) ;
}

class Host : public Employee
{
   private:
        char duty[70] ;
   public:
        Host( ) ;
        Host(int, char*, char*) ;
        void setDuty(char*) ;
        char* getDuty( ) ;
}
```

```
class Time
{
   private:
         int hours ;
         int minutes ;
         int seconds ;

   public:
         Time(int, int, int) ;
         //setters
         //getters
}

class Flight
{
   private:
         char sourceCity[20] ;
         char destinationCity[20] ;
         Time departureTime ;
         Time landingTime ;
   public:
         Flight(char*, char*, Time, Time) ;
         //setters
         //getters
}
```

Good Luck