EECS1720
Worksheet 2 – Class Basics + Aggregation & Composition - SOLUTION

1) What is the HAS-A relationship between two objects?

   a) **a relationship where one object "is associated" another object B.**
   b) a relationship between abstractions wherein one class is a subclass of another class C.
   c) a relationship between the type of an object and an instance of the type D. d) something else

2) What is aggregation?

   a) **a relationship where one object A "is associated" with another object B, however B may exist independently from the object A**
   b) a relationship where an object A is a specialized "version" of another object B, and assumes all attributes and fields from the other object B
   c) a relationship where one object A "belongs to" another object B, and A may only exist as a part of B
   d) something else

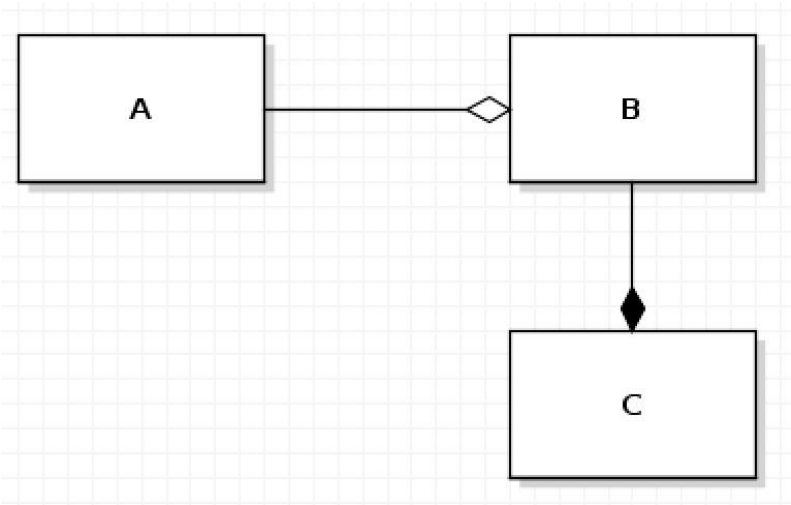
3) Why is composition sometimes necessary?

   a) To ensure that an object remains fully accessible to a client and may be modified whenever, and however the client sees fit
   b) **To avoid unwanted sharing of memory between different reference variables that a client may have access to in an application**
   c) To ensure that an object does NOT have ownership over another object
   d) None of the above

4) A utility class is a class that:

   a) Has only methods defined
   b) Has only fields defined
   c) **Has only static features (methods/fields)**
   d) Has only non-static features (methods/fields)

5) The diagram below describes several relationships between classes, which of the following best describes these relationships?



a) A is an aggregation of B, and C is a composition of B
b) A is a composition of B, and C is an aggregation of B
c) B is an aggregation of A and an aggregation of C
d) B is an aggregation of A and a composition of C
e) C is an aggregation of B, and B is a composition of A
f) **C is a composition of B, and B is an aggregation of A**

6) A custom constructor

a) Is a constructor that accepts as its argument, another object of the same class that is currently being constructed
b) **Is a constructor that accepts any arbitrary (non-empty) list of arguments**
c) Is a constructor that has no arguments
d) None of the above

7) If a field or method has a private access modifier, then:

a) The field/method is accessible within the constructors of the class in which it is defined, but nowhere else
b) The field/method is not directly accessible to a client, however it is accessible to any method within the class in which it is defined
c) **The field/method is not directly accessible to a client, however it is accessible to any method or constructor within the class in which it is defined**
d) The field/method is accessible to any constructor or method within the class in which it is defined, and externally to any client using the class

8) An overloaded method is a method that:

a) **has the same name, but different signature as another method in the same class**
b) has the same signature, but different name as another method in the same class
c) returns more than one value
d) performs the same function as a constructor within the class

9) The term "encapsulation" refers to:

a) The fact that an object can be instantiated from a class
b) **The bundling of both data and methods together into a single unit (instantiated as an object, where the methods are configured to allow/ not allow access to fields)**
c) The concept of associating more than one class together into a single object
d) Giving a client direct access to the modify the fields of an object without using methods defined for that class

Implementing Classes

10) Choosing fields:

For each of the following kinds of values, choose appropriate fields to represent the value (imagine you are trying to implement a class that represents the value). Try to come up with two alternate sets of fields that could represent each kind of value.

a) weight

```
double  kilograms;          float kg;
double  pounds;             float lb;
```

b) temperature

```
double celcius;
double farenheight;
```

c) time of the day

```
int hour;          in min;          int seconds;    //OR
double hours;      double minutes;  double secconds;
```

d) day of the year

```
int dayOfMonth;
int monthOfYear;
String dayOfWeek;
```

11) Default constructor:

a) Suppose that a Temperature is represented as a floating point value in degrees Celcius. Implement a default (no argument) constructor.

```
public Temperature( ) {

        this.celcius = 22.0f;          // room temp as default

}
```

b) Suppose that a TimeOfDay is represented as an integer hour and an integer minute. Implement a default (no argument) constructor.

```
public TimeOfDay( ) {

        this.hour = 12;          // choose any defaults
        this.minute = 0;

}
```

12) Custom constructor:

a) Suppose that a Temperature is represented as a floating point value in degrees Celcius. Implement a custom constructor that initializes the temperature given a value in degrees Celcius

```
public Temperature(float celcius) {

        this.celcius = celcius;

}
```

b) Suppose that a TimeOfDay is represented as an integer hour and an integer minute. Implement a custom constructor that initializes a time given an hour and a minute.

```
public TimeOfDay(int hour, int minute) {

        this.hour = hour;
        this.minute = minute;

}
```

13) Copy constructor:

a) Suppose that a Temperature is represented as a floating point value in degrees Celcius. Implement a copy constructor that initializes the temperature given another Temperature reference.

```
public Temperature(Temperature other) {

        this.celcius = other.celcius;                // alias (aggregation)

}
```

b) Suppose that a TimeOfDay is represented as an integer hour and an integer minute. Implement a copy constructor that initializes a time given another TimeOfDay reference.

```
public TimeOfDay(TimeOfDay other) {

        this.hour = other.getHour();        // assumes public accessors exist
        this.minute = other.getMinute(); // otherwise use other.hour,
                                                    // other.minute if hour and
                                                // minute are public fields

}
```

14) Implement a set method:

a) Suppose that a Temperature is represented as a floating point value in degrees Celcius. Implement a set method that sets the value of a temperature given a value in degrees Celcius.

```
public void setCelcius(float celcius) {

        this.celcius = celcius;

}
```

b) Suppose that a TimeOfDay is represented as an integer hour and an integer minute. Implement a set method that sets the value of a time given an hour and a minute.

```
public void setTimeOfDay(int hour, int minute) {

        this.hour = hour;
        this.minute = minute;


}
```