Class test 4 Answers

Question 1 ..... (11 marks)

The base TV class is given by the following class definition.

```
class TV {
public:
  TV();
  TV(string, string, int, int);
  bool pushOnOffButton();
  int pushChannelButton();
  virtual string getStatus() = 0;
  virtual string getDimensions() = 0;
  virtual ~TV();
protected:
  int getChannel();
  bool getOnOffStatus();
  virtual void toggleOnOff();
  virtual void changeChannel();
  void calculateDimensions(int);
  float getWidth();
  float getHeight();
private:
  string brand;
  string model;
  bool isOn;
  int currentChannel;
  int maxChannels;
  float width; // in inches
  float height; // in inches
};
```

1.1 The implementation of the `pushOnOffButton` and `pushChannelButton` is given.

```
bool TV::pushOnOffButton() {
  toggleOnOff();
  return getOnOffStatus();
}
```



```
int TV::pushChannelButton() {
  changeChannel();
  return getChannel();
}
```

```
int TV::pushChannelButton() {
  changeChannel();
  return getChannel();
}
```

a) Is this an implementation of the Template Method design pattern? Provide an explanation. (2)

b) Write the `toggleOnOff` and `getOnOffStatus` functions. (4)

1.2 Consider the implementation of a constructor defined in the TV class. (3)

```
TV::TV(string brand, string model, int maxChannels, int size) {
  this->brand = brand;
  this->model = model;
  isOn = false;
  this->maxChannels = maxChannels;
  currentChannel = 0;
  calculateDimensions(size);
}
```

The last parameter - `size` - is an integer value which represents the diagonal length (s) of the TV and is given in inches. The TV class, however, does not store the diagonal length of the TV but the width (w) and height (h) of a TV. Write the code for `calculateDimensions`. The formulae for h and w, given s, are:

$$h = \sqrt{\frac{s^2}{4.16}}$$

$$w = 1.777778 * h$$

1.3 Specific TV brands inherit from TV and provide implementations of functions specific to them. (2)

1.1.a No, because all the functions that are called are not virtual and are implemented in TV class

1.1 b) void TV::toggleOnOff(){
this->isOn = !isOn; }

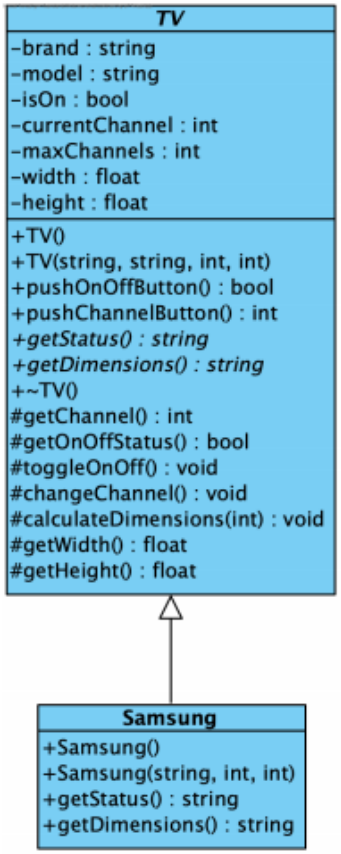
bool TV::getOnOffStatus(){
return this->isOn; }

1.2 void TV::calculateDimensions(int s){

this->height = sqrt(pow(s,2)/4.16));

this->width = 1.777778 * this->height; }

1.3 string getStatus()
string getDimensions()



How would you apply the Composite design pattern to model the wall of TV's? To answer this question, answer the questions that follow.

2.1 For the Composite design pattern, which participants of the pattern do `TV` and `Samsung` represent? (2)

2.2 You may assume that a function exists to calculate the optimal number of TV's required to cover the wall. The function, `calculateWallDimensions`, accepts as a parameter the diagonal length (`size`) of the wall in inches and returns a triple. The first element of the triple is the number of TV's that make up the width of the wall TV's, the second is the number of TV's making up the height. The final element of the triple is the optimal TV size required, defined in inches. These three values must be stored by an object of class `Wall`. (6)

When a wall of TV's is defined, no TV's are assigned to the wall until a member function, `createWall`, is called. This function, returns a boolean if the creation was successful. You may assume, for now, the wall is made of Samsung TV's.

Design a class, call it `Wall`, that when included with the above class diagram, will complete the implementation of the Composite design pattern. Provide only the class definition.

Make use of the `tuple` defined in C++11 for the return type of the `calculateWallDimensions` function. The `tuple` must take 3 `int` values as template parameters, that is `tuple<int,int,int>`.

2.3 Does it make sense to program the Iterator design pattern to iterate through the wall of TV's? Provide reasons for your answer. (2)

2.1 TV: Component

Samsung: Leaf

2.2.

```
class Wall : public TV
```

```
{
private:
   int numTVsWidth;
   int numTVsHeight;
   int optimalTVSize;
   TV *tvs[];
public:
   Wall();
   Wall(string huntMethod, string speciality);
getSize();
getHeight();
getWidth();
   ~Wall();
   tuple<int, int, int> calculateWallDimensions(int inches);
   bool createWall(TV tvs[]);
```

2.3. Yes, as it would provide a way to iterate through the TV objects outside of the Wall class, making it easier for the TVs to be traversed in a uniform manner.

Question 3 ..... (10 marks)

Consider the following main program and draw the UML Sequence diagram showing the function calls over time. Include all calls in the `myTV` object for which you have been given the code. That is, for `pushOnOffButton`, `pushChannelButton` and the constructor that takes parameters.


```
int main() {

  TV* myTV = new Samsung("UA32N5003BRXXA",99,32);

  myTV->pushOnOffButton();
  myTV->pushChannelButton();

  cout << myTV->getStatus() << endl;
  cout << myTV->getDimensions() << endl;

  delete myTV;

  return 0;
}
```

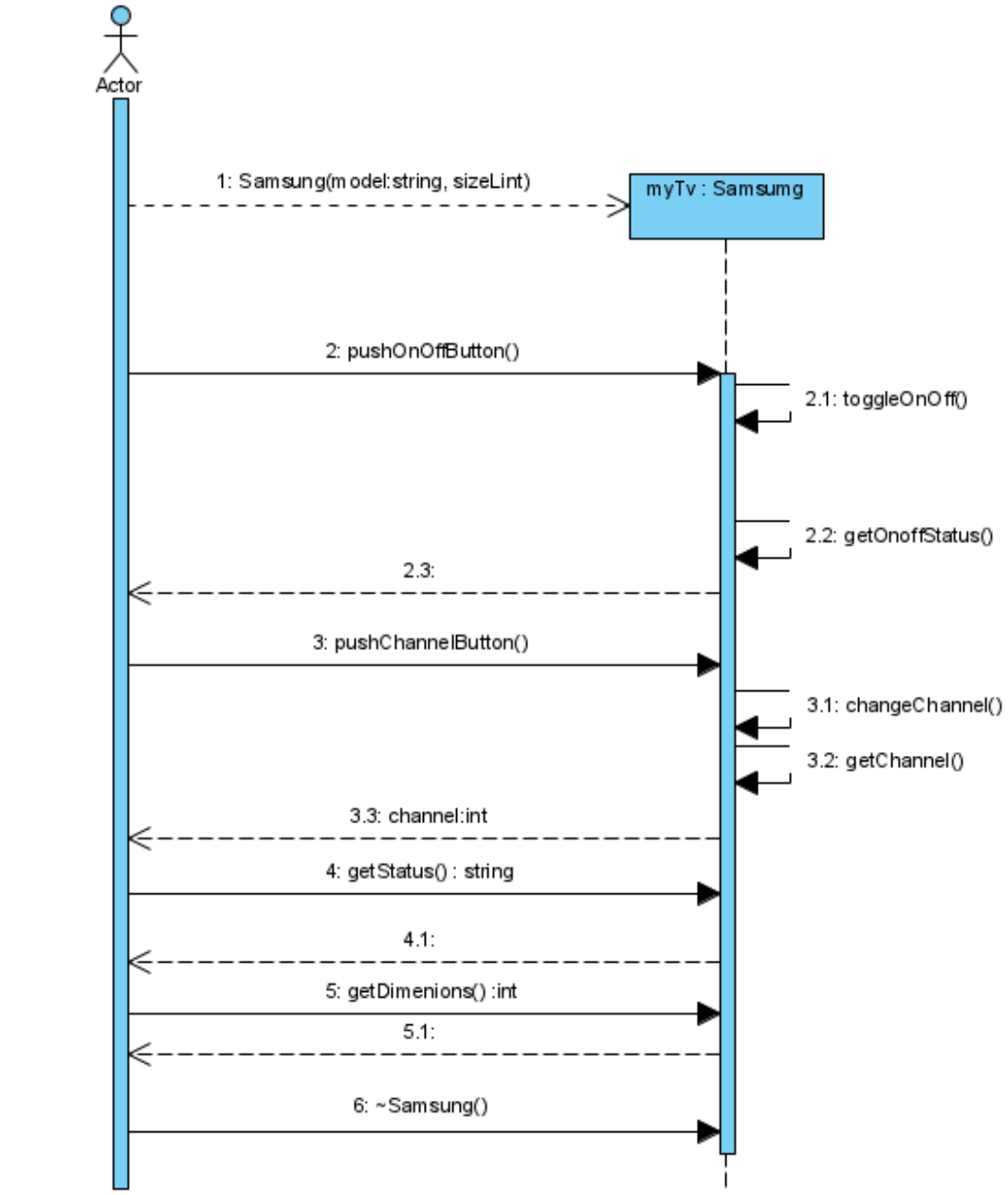
Question 4 ..... (9 marks)

Most TV's allow you to manipulate the brightness, contrast, volume and other aspects of the TV other than the on-off switch and the channels.

4.1 Which pattern would you use to add this type of functionality to individual TV's and the wall of TV's? (1)

4.2 Provide a high-level UML class diagram design showing how this additional functionality can be incorporated into the initial Composite design. In your high-level design, show only the classes and their relationships. (8)

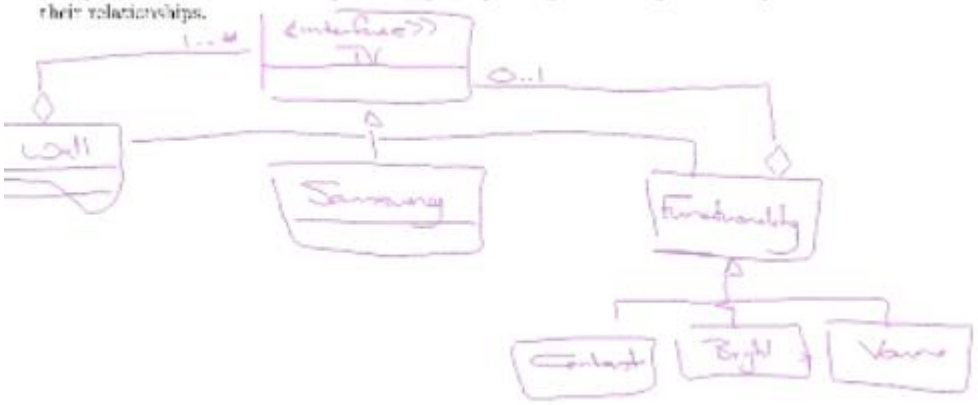
3.



4.1. Decorator

4.2.

incorporated into the initial Composite design. In your high-level design, show only the classes and their relationships.



5.2 Observer