Q1:

Adapter design pattern is a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ design pattern

Creational

Structural ✓

Behavioral

Q2:

In C++, an abstract class is defined by the keyword:

abstract

interface

struct

None of the Above ✓

Q3:

Abstract Factory design pattern is a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ design pattern

Creational ✓

Structural

Behavioral

Q4:

UML is specifically designed to work with Java only

True

False ✓

Q5:

See the definition below:

class GameSetting{

static GameSetting\* \_instance;

public:

GameSetting(){}

static GameSetting\* getInstace() {

if(\_instance == NULL)

\_instance = new GameSetting();

return \_instance;

}

};

GameSetting \* GameSetting::\_instance = NULL;

**Is this correct implementation of a Singleton?**

True

False ✓

Q6:

When designing, you should always prefer composition to inheritance

True ✓

False

Q7:

Whenever you create a reference to an object, it is called a delegate

True

False ✓

Q8:

Design patterns are generalized solutions to generalized problems that occur with some modicum of frequency when you're creating software using the functional programming paradigm.

True

False ✓

Q9:

What is not a type of Design Pattern?

Creational

Behavioral

Structural

Adaptable ✓

Q10:

Based on the two principles of OOP, we should always define to an interface and not implementation.

True ✓

False