DPW - Written Exam NAME: Kevin Fernandez

Define the following design pattern in detail. Make sure to include the pattern classification(type).

Prototype: Prototype is a creation design pattern that focuses on cloning the state and behavior of any given value. It is similar to classes that it is used to create a new object in which the program can interact with, but unlike classes that serve as a blueprint. The prototype can override any of its methods and variable making the final product change. This happened because of Prototype Inheritance that doesn't create a blueprint but a prototype.

Singleton: Singleton is creation design pattern that its purpose is to make sure a class only has one object on run time. This is useful for object that is needed to be the same throughout the application and running more that one instance may affect the quality of the application.

Factory: Factory is a creational design pattern that works within a static method that ensures the object is created (instantiated) at a later time or when a user decides.
This design pattern is also great for scalability and maintainability since this behavior is saved inside of a method.

```
12  var Dog = (function(){
13      function Dog(){
14          console.log("Dog Created");
15          this.name = "Fido";
16      }
17      function bark(){
18          return "I am barking...";
19      }
20      return Dog;
21  });
```

Assume the above code is employing the Prototype Pattern. Identity the **one** error in the pattern and provide the solution

Problem: The bark function is not part of the prototype, and it could cause problem with the behavior of the prototype even is with in closure. Also it doesn't share though inherence.
Solution: use Dog.prototype.bark = function() {
Return "I am barking…";
}

```
14  class ToyFactory{
15      constructor()
16      {
17
18      }
19      createToy(name)
20      {
21          if(name == "sitnspin")
22          {
23              var s =  new SitNSpin();
24          }
25          else if(name == "pollypocket")
26          {
27              var p = new PollyPocket();
28          }
29          else
30          {
31              throw "Attempting to create "+ name +" factory can't handle it!";
32          }
33      }
34  }
```

Assume the above code is employing the Factory Pattern. Identity the **two different types** of errors in the pattern and provide the solutions (Total of 3 errors, two being the same type)

Problem: createToy is not static and it requires to be instantiated removing one of the feature of factory design pattern.

Problem: There is no retuning the value once it receive the argument.

Solution: make createToy static so in can be call from the class itself and not a instance of the class.

Solution: Have one variable that will hold the object created depending of the argument that is pass in and return it after the last logic structure.

```
3   class DPW_exercise {
4       constructor() {
5           console.log("singleton created");
6       }
7       static getInstance() {
8           if (!DPW_exercise._instance) {
9               DPW_exercise._instance = new DPW_exercise();
10          } else {
11              throw "wtf are you doing"
12          }
13      }
14  }
15
16  var myDWP_exercise = new DPW_exercise();
17
```

Assume the above code is employing the Singleton Pattern. Identity the **two** errors in the pattern and provide the solutions

Q:Write 3 lines of code using JavaScript that will create a custom event called “modelcomplete”, attach a property called “data” on the DTO and also have it dispatched on an object called “node”.

A:
Const complete = new Event(“modelcomplete”);
complete.data = data;
document.dispatchEvent(complete);

Q:What is the difference between Composition and Aggregation.

A: Composition is an Object-Oriented term that expresses a Has-A type of relationship with a class to the other class(sometimes multiple) A example of this is a car, a car has an engine to help it turn on but an engine itself is not a car but a part of it (important part) or it defines the car but, complete it.

Aggregation is an Object-Oriented term that expresses an Is-A type of relationship meaning that inherent's some of the traits of its base class and it's not easily changeable. An example of this is, Car is a vehicle meaning that it is similar but not the same as a bus. Making vehicles a parent of both car and bus.