1. (ქულა: 3) რა სხვაობაა Class და Interface-ს შორის?

Class არის თითქოს ნახატი ან გეგმა, რომლითაც ობიექტებს ვქმნით. მას შეუძლია ჰქონდეს ატრიბუტები და მეთოდები, რომლებიც უკვე გაწერილია და მუშაობს.
Interface კი მხოლოდ გეუბნება რა მეთოდები უნდა ჰქონდეს კლასს, მაგრამ არ ეუბნება როგორ უნდა იმუშაოს. ეს მხოლოდ დავალებაა, რომელიც კლასმა თვითონ უნდა შეასრულოს.

2. **(ქულა**: 3**)** რა სხვაობაა Public, protected და private-ს შორის?

ეს არის "ხილვადობის" დონეები. ვის რა შეუძლია ნახოს:
Public - ყველას ხედავს, ყველგან.
Protected - ხედავს მხოლოდ ეს კლასი და მისმა შთამომავლები (მემკვიდრე კლასები).
Private - ხედავს მხოლოდ ეს კლასი თვითონ.

3. **(ქულა**: 3**)** შესაძლებელია თუ არა Interface-ის ობიექტის შექმნა?

არა, პირდაპირ ვერ შექმნი. Interface არის მხოლოდ "გეგმა", რომელსაც არ აქვს რეალური იმპლემენტაცია.
მაგრამ შეგიძლია შექმნა ობიექტი იმ კლასისგან, რომელიც ამ Interface-ს ახორციელებს, და ცვლადი იყოს Interface ტიპის.

4. შესაძლებელია თუ არა Abstract Class-ის ობიექტის შექмნა?

არა, Abstract Class-ისგან ვერ შექმნი ობიექტს პირდაპირ. ის არის "არასრული" კლასი, რომელიც გამიზნულია იმისთვის, რომ სხვა კლასები მემკვიდრეობით მიიღონ და დაასრულონ.
რჩება კონკრეტული (არა-abstract) შვილი კლასი შექმნა, რომელიც მემკვიდრეობით მიიღებს abstract კლასს და დაასრულებს ყველა abstract მეთოდს.

5. **(ქულა**: 6**)** დაწერეთ კოდი Singlton Design Pattern-თვის.

```
public class Singleton {
   private static Singleton instance;

   private Singleton() {}

   public static Singleton getInstance() {
```

```
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }

    public void doSomething() {
        System.out.println("Singleton მუშაობს!");
    }
}

// გამოყენება:
// Singleton obj = Singleton.getInstance();
```

6. **ქულა**: 6) დაწერეთ კოდი Factory Design Pattern-თვის.

```
interface Animal {
    void makeSound();
}

class Dog implements Animal {
    public void makeSound() {
        System.out.println("ყეფა");
    }
}

class Cat implements Animal {
    public void makeSound() {
        System.out.println("მიაუ");
    }
}

class AnimalFactory {
```

```
    public static Animal createAnimal(String type) {
        if (type.equalsIgnoreCase("dog")) {
            return new Dog();
        } else if (type.equalsIgnoreCase("cat")) {
            return new Cat();
        }
        return null;
    }
}

// გამოყენება
// Animal myPet = AnimalFactory.createAnimal("dog");
// myPet.makeSound();
```