SNUS – Primer prvog kolokvijuma

1. Napisati konzolnu aplikaciju koja omogućava simulaciju i nadgledanje punjenja i pražnjenja silosa. **Uputstvo**:
   - Kreirati klasu **Silo** koja sadrži svojstva **int Level** i **string Status. Level** predstavlja trenutnu vrednost nivoa žitarica u silosu, a status nam govori da li je silos prazan, polupun ili pun. Nivo silosa je na početku **2 m**, a status je **polupun**. Ove inicijalne vrednosti postaviti u okviru konstruktora.
   - Kreirati delegat **SiloLogHandler** prema potrebama zadatka.
   - Kreirati klasu **SiloMonitoring** sa poljima **Silo silo**, **Thread thread**, događajem **SiloLogHandler siloStatusChanged**, kao i metodama koje slede.
   - Kreirati konstruktor za klasu **SiloMonitoring**, koji prima parametar **Silo silo**, i inicijalizuje sva polja ove klase.
   - Kreirati metodu **public void Start(int seconds, int changeRate)** koja pokreće nit **thread**, odnosno punjenje i pražnjenje silosa. Ulazni parametri metode predstavljaju period nakon kojeg će se vrednost nivoa u silosu izmeniti i veličinu te promene.
   - Omogućiti prosleđivanje pomenutih parametara do niti **thread.**
   - Kreirati metodu **private void Simulation(object obj)**, koja vrši punjenje i pražnjenje silosa u okviru niti **thread**. Silos se puni do nivoa od **20 m**, a zatim se prazni do **0 m**. Na **20 m** status silosa postaje **pun**, a na **0 m** postaje **prazan**. Događaj **siloStatusChanged** se poziva samo u ova dva slučaja.
   - Kreirati metodu **private void OnSiloStatusChangedConsole(string status)**, koja ispisuje poruku o trenutnom stanju silosa na konzolu, a poziva se kada se status silosa promeni, odnosno pozove događaj **siloStatusChanged**.
   - Kreirati metodu **private void OnSiloStatusChangedFile(string status)**, koja ispisuje poruku o trenutnom stanju silosa u fajl **siloLog.txt**, a poziva se kada se status silosa promeni, odnosno pozove događaj **siloStatusChanged**. Koristiti klasu **StreamWriter** i metodu **File.AppendText(String)**.
   - Bez korišćenja **Console.ReadKey()** metode sprečiti završavanje **Main** niti pre pražnjenja silosa i dodatne pauze od **5 sekundi**.

2. Napisati konzolnu aplikaciju koja omogućava simulaciju rada baze podataka vozača i automobila. Potrebne klase i interfejsi su dati na narednoj strani. **Uputstvo:**
   - Implementirati metodu **GetExperiencedDrivers()** koja vraća listu svih vozača sa preko **10** godina vozačkog iskustva.
   - Implementirati metodu **GetOldtimers()** koja vraća listu svih automobila starijih od **30** godina, a čiji je proizvođač **“Jaguar”, “Rolls-Royce”** ili **“Aston Martin”**.
   - Implementirati statički konstruktor klase **CombinedDataBase**, unutar kojeg će biti inicijalizovane baze podataka vozača i automobila. Vozače i automobile učitati iz priloženog XML fajla.
   - Izraditi izveštaj koji vraća spisak automobila za svakog vozača.
   - Izraditi izveštaj koji vraća ime vozača za svaki od automobila.
   - Izraditi izveštaj koji vraća listu neiskusnih vozača, čije ime počinje na slovo P i poseduju bar dva oldtajmera.

```
public class Driver
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int YearsDriving { get; set; }
}

public class Car
{
    public int DriverId { get; set; }
    public string Manufacturer { get; set; }
    public int Age { get; set; }
}

public class DriverDataBase
{
    public List<Driver> Drivers { get; set; }
}

public interface IDDBUpgrade
{
    List<Driver> GetExperiencedDrivers();
}

public class UpgradedDDB : DriverDataBase, IDDBUpgrade
{
    //Implementirati interfejs IDDBUpgrade
}

public class CarDataBase
{
    public List<Car> Cars { get; set; }
}

public interface ICDBUpgrade
{
    List<Car> GetOldtimers();
}

public class UpgradedCDB : CarDataBase, ICDBUpgrade
{
    //Implementirati interfejs ICDBUpgrade
}

public static class CombinedDataBase
{
    public static UpgradedDDB DDB;
    public static UpgradedCDB CDB;

    static CombinedDataBase() { }

    //Izvestaji
}
```