VOP F18 Tællende Aktivitet 2 16. april 2018

Denne tællende aktivitet afvikles i øvelsestimerne under lektion 8.

Opgaven skal løses individuelt i tidsrummet 10:15 – 11:15 (1. time af øvelses timerne) under lektion 8.

Det er ikke tilladt at:

- Kommunikere med andre end instruktorerne under afviklingen.
- Benytte Internettet til andet end hentning og aflevering af opgaven

Der skal oprettes et NetBeans-projekt af typen ***JavaFX FXML Application*** som navngives "abcdeXX Tael2 F18", hvor "abcdeXX" er den studerendes SDU brugernavn.

Aflevering skal ske på BlackBord->Assignments senest kl. 11:15.

Hele NetBeansprojektet skal uploades i en zip-fil, dannet ved at markere projektet i *Projects*-vinduet og fra *File*-menuen vælge *Export Project->To ZIP…* Sørg for at den dannede zip-fil er navngivet korrekt og har extension zip ("abcdeXX Tael2 F18.zip").

NB: Hvis navne konventionerne ikke er overholdt, trækkes op til 2 point af de 15 mulige!

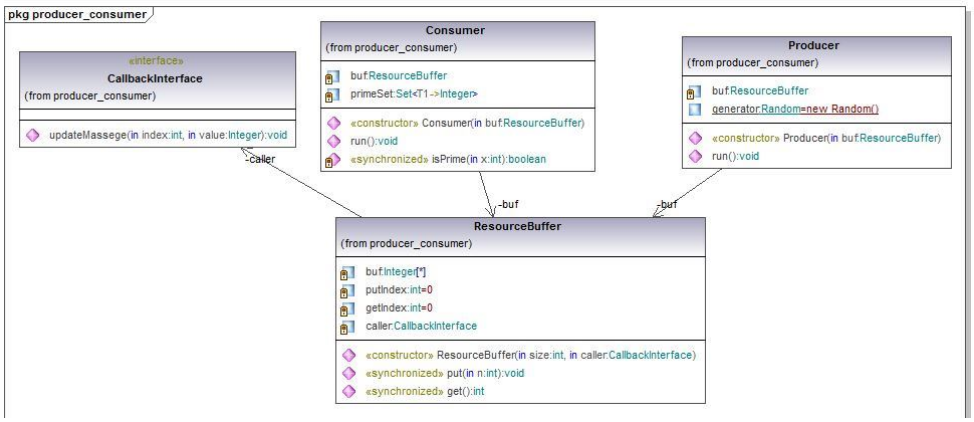
Opgaven: Producer/Consumer med javaFx brugerflade. 15 point

Denne opgave tager udgangspunkt i de trådnings -eksempler og -opgaver, vi behandlede i lektion 6 og 7 (*CirkularBuffer* og eksemplerne på *opdatering af javaFx brugerflade fra en anden tråd*).

Hent producer_consumer.zip og pak filen ud i projektets src-mappe. Pakken indeholder:

- Interfacet `CallbackInterface` som definerer metoden `updateMessage(int index, Integer value);` Interfacet skal benyttes til at opdatere brugerfladen i opgave b.
- Kodeskelet til klassen `ResourceBuffer`. Til test af opgave a kan denne implementation benyttes fra `main()`-metoden:

```
CallbackInterface caller = new CallbackInterface(){
    @Override
    public void updateMessage(int index, Integer value) {
        System.out.println(index + " er opdateret med " + value);
    }
};
ResourceBuffer rb = new ResourceBuffer(5, caller);
```



`package producer_consumer` ønskes implementeret ud fra ovenstående diagram.

Opg A Producer/Consumer uden brugerflade (9 point)

`Producer.java` skal implementere `Runnable`-interfacet og i `run()` metoden producere *tilfældige heltal* i intervallet `[10..99]`. De producerede tal lægges ind i en fælles buffer.

`Consumer.java` skal også implementere `Runnable`-interfacet og hente tallene ét ad gangen fra den fælles buffer. Det skal undersøges om hvert hentet tal er et primtal. I så fald tilføjes tallet til et *sorteret* `Set<Integer>`.

Benyt fx denne algoritme til tjek af primtal:

```
private synchronized boolean isPrime(int x) {
    int sqrt = (int) Math.sqrt(x) + 1;
    for (int i = 2; i < sqrt; i++) {
        if (x % i == 0) {
            return false;
        }
    }
    return true;
}
```

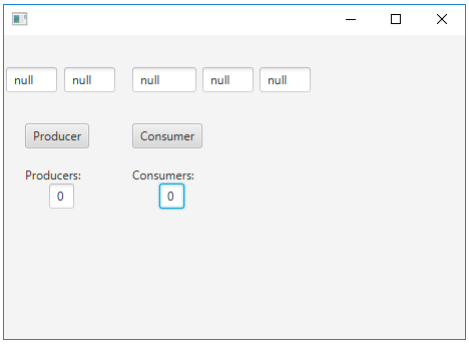
Når en consumer har opsamlet 5 forskellige primtal, udskrives de på `System.out`, hvorefter den tråd consumer'en eksekveres i stopper.

Begge klasser modtager den samme instance af `ResourceBuffer`, som parameter til constructoren, og skal sove et *tilfældigt antal millisekunder*, mellem 100 og 300ms, mellem hvert tal der genereres/hentes.

`ResourceBuffer.java` indeholder den fælles ressource, som `Producer` og `Consumer` skal tilgå. Klassen skal indeholde:

- Et array af `Integer` – typen (den fælles buffer) med den længde, der angives i constructoren (sæt længden til 5 i disse opgaver). Ved start skal alle pladser i arrayet have værdien `null`.
- To indeks-variable, til styring af hvor i arrayet værdier skal indsættes af `Producer` og hentes af `Consumer`. Begge initialiseres til `0`.
- Metoderne `void put(int n)` og `int get()`til indsætning og hentning af tal fra arrayet. Der skal benyttes synkronisering, så:
    - Når en `Producer` indsætter et tal, kan det kun placeres i arrayet, hvis pladsen er `null`. Ellers skal der ventes indtil en consumer har fjernet det tal der står der.
    - Når en `Consumer` forsøger at hente et tal, kan det kun ske hvis pladsen ikke er `null`. Ellers ventes indtil en producer har indsat et tal.
    - Arrayet skal gennemløbes cirkulært af begge indeks variable, startende på indeks = 0.
    - Når der er indsat/fjernet et tal, skal `updateMessage(int index, Integer value)`fra `CallbackInterface` kaldes med aktuelle indeks i arrayet og den nye værdi (et heltal eller `null`).
- En constructor med signaturen
  `public ResourceBuffer(int size, CallbackInterface caller).`
  hvor `size` er størrelsen på arrayet og `caller` er en implementation af interfacet.
- En `main()` metode til test. Der kan fx oprettes 2 tråde med `Producer`s og 3 `Consumer`s. Sørg for at alle producere stopper når programmet afsluttes.

Opgave B Integration med javaFx (6 point)



Brugerfladen skal indeholde:

5 `TextFields` eller `Labels` som viser det øjeblikkelige indhold af `ResourceBufferen`.

En knap til at starte tråde, som eksekverer `Producers` og et `TextField` el. `Label`, som viser hvor mange der er startet.

En knap til at starte tråde, som eksekverer `Consumers` og et `TextField` el. `Label`, som viser hvor mange der er startet.

`FXMLDocumentController-klassen:`

De 5 `TextFields` skal løbende opdateres fra `ResourceBufferen` ved kald til en implementation af `CallbackInterface.` Dette kan fx gøres ved at erklære et array `TextField[5](`eller `Label[5]),` indeholdende de 5 felter. Implementer `CallbackInterface,` enten som en indre klasse, eller som en variabel med anonym implementation af `updateMessage()`.