გამოსახულება (პოლიმორფიზმი) - 50 ქულა

თქვენი ამოცანაა მოიფიქროთ არითმეტიკული გამოსახულებების აღმწერი ინტერფეისი და მისი შვილობილი კლასები. ამ კლასების გამოყენებით შესაძლებელი უნდა იყოს არითმეტიკული გამოსახულებების აღწერა და მისი მნიშვნელობის დათვლა. გამოსახულებები უნდა მუშაობდეს ინტეჯერებზე და იყენებს მხოლოდ +, -, * და / ოპერატორებს. მაგალითად შესაძლო გამოსახულებებია:

- 1 + 2
- 1 + 2 * 3
- (1 + 2) * 3

მოცემულ Expression ინტერფეისში უნდა აღწეროთ თუ რა მეთოდები უნდა ქონდეს მას. ასევე დაამატეთ ამ ინტერფეისი ყველა საჭირო შვილობილი კლასი თავიანთი იმპლემენტაციებით.

ასევე მოგეთხოვებათ იმპლემენტაცია გაუკეთოთ ExpressionFormatter.format მეთოდს რომელსაც გადაეცემა Expression ინტერფეისის ობიექტი და უკან უნდა აბრუნებდეს შესაბამისი არითმეტიკული გამოსახულების რეპრეზენტაციას როგორც String ტიპის მნიშვნელობას.

ამ კლასების იმპლემენტაციისას არსად არ გამოიყენოთ branch statement-ები როგორიცაა if/else.

რიგი (ნაკადები) - 50 ქულა

თქვენი ამოცანაა იმპლემენტაცია გაუკეთოთ **ProducerConsumerQueue** კლასს შემდეგი ინტერფეისით, რომელიც უნდა იყო Thread Safe (ანუ მრავალი ნაკადიდან მისი გამოყენება პრობლემებს არ უნდა ქმნიდეს):

- ProducerConsumerQueue(int capacity) - კონსტრუქტორი რომელიც არგუმენტად იღებს დროის ნებისმიერ მომენტში მაქსიმუმ რამდენი ელემენტი შეიძლება იყოს შენახული რიგში.
- put(T value) - რომელიც არგუმენტად გადაცემულ მნიშვნელობას ამატებს რიგში. თუ რიგი სავსეა ელოდება ადგილის გამოთავისუფლებას და მხოლოდ შემდეგ ამატებს ელემენტს.
- T take() - რომელიც რიგიდან იღებს ყველაზე ძველად დამატებულ ელემენტს და უკან უბრუნებს კლიენტს. თუ რიგი ცარიელია ელოდება ერთი მაინც ელემენტი დაემატოს რიგში.

მაქსიმალური ქულის ასაღებად java.util.Concurrent პაკეტიდან შეგიძლიათ გამოიყენოთ მხოლოდ Lock/ReentrantLock, Condition. სხვა კლასების გამოყენების შემთხვევაში თქვენი ნამუშევარი შეფასდება მაქსიმუმ 50%-ით.

სტრიმები - 40 ქულა

იმპლემენტაცია გაუკეთეთ **Streams** კლასში აღწერილ ორ ფუნქციას:

- **List<T> removeEveryNth(Stream<T> items, int n) -** გადმოცემული სტრიმიდან უნდა ამოაგდოს ყოველი n-ური ელემენტი და დანარჩენელი ელემენტები დააბრუნოს უკან სიის სახით. მაგალითად [1, 2, 3, 4, 5] -დან ყოველი მეორეს ამოგდებით მივიღებთ: [1, 3, 5]
- **List<T> removeConsecutiveDuplicates(Stream<T> items, Comparator<T> cmp) -** გადმოცემული სტრიმის ერთმანეთის შემდგომი ერთნაირი ელემენტებიდან მხოლოდ ერთი უნდა დატოვოს. ელემენტების შედარებისთვის უნდა გამოიყენოთ გადმოცემული შემდარებელი **cmp** არგუმენტი. მაგალითად [1, 2, 2, 1, 2, 2, 2] უნდა აქციოს: [1, 2, 1, 2] -ად.

მაღაზია (ვებ არქიტექტურა) - 60 ქულა

თქვენი ამოცანაა მოიფიქროთ ინტერნეტ მაღაზიის ვებ არქიტექტურა. მოთხოვნები იხილეთ ქვემოთ.

მომხმარებლების ტიპები

1. რიგითი მომხმარებლები - რომლებსაც შეუძლიათ საიტის დათვალიერება და ნივთების შეკვეთა.
2. გამყიდველები - მათი ანგარიში რეგისტრაციის შემდეგ მოითხოვს ადმინისტრატორისგან გადამოწმებას და გააქტიურებას. გააქტიურებამდე ასეთ მომხმარებლებს არ შეუძლიათ საიტზე პროდუქციის გამოქვეყნება.
3. ადმინისტრატორები - რომლებსაც ევალებათ ახლად დარეგისტრირებული გამყოდველების ინფორმაციის გადამოწმება და შესაბამისად ანგარიშის გააქტიურება ან დაბლოკვა.

მაღაზიის ფუნქციონალი:

1. საიტზე უნდა ქონდეს პროდუქტების გატეგორიების ცნება
2. კატეგორიების დამატება/წაშლა მხოლოდ ადმინისტრატორს უნდა შეეძლოს
3. გამყოდველებს უნდა შეეძლოთ კატეგორიაში ახალი პროდუქტი დაამატონ და თან მიუთითონ: სახელი, ფასი, რაოდენობა, მოკლე აღწერა
4. რიგით მომხმარებლებს უნდა შეეძლოთ მაღაზიის დათვალიერება კატოგირიების მიხედვით, პროდუქტების კალათაში დამატება და ბოლოს შესყიდვა.
5. რიგით მომხმარებლებს უნდა შეეძლოთ უკვე ნაყიდ პროდუქტზე შეფასების დაწერა. შეფასებები პროდუქტის გვერდზე უნდა ჩანდეს.
6. ადმინისტრატორს უნდა შეეძლოს ნებისმიერი მომხმარებლის (რიგითი თუ გამყიდველი) ან რომელიმე პროდუქტი დაბლოკოს საიტიდან.

გაითვალისწინეთ რომ ვებსაიტი დაცული უნდა იყოს, ანუ ადმინისტრირების ფუნქციონალი მხოლოდ ადმინისტრატორს უნდა შეეძლოს გამოიყენოს და გაკეთებული ჯავშნის სხვისთვის გადაცემა მხოლოდ ჯავშნის ავტორს.

ნამუშევრის ორგანიზება:

- **db.txt** - ფაილში ჩაწერეთ თუ რა ცხრილები დაგჭირდებათ ბაზაში და თითოეულ ცხრილს რა სქემა ექნება. გამართული SQL-ით აღწერა არ მოგეთხოვებათ, საკმარისია თითოეული ცხრილისთვის ჩამოთვალოთ ველების სახელები, ტიპები, primary და foreign გასაღებები.
- **urls.txt** - ფაილში ჩაწერეთ თუ რა URL-ებს მოემსახურება თქვენი იმპლემენტაცია და ყოველ URL-ს მიუთითეთ თუ რომელი სერვლეტი მოემსახურება მას.
- თითოეულ სერვლეტს მოკლედ მიუთითეთ თუ რა ფუნქციონალს უკეთებს იმპლემენტაციას და როგორ. როგორში იგულისხმება ბაზასთან კავშირს როგორ ამყარებს, სესიაში ინახავს თუ არა რამეს, თავს როგორ იცავს ანუ მომხმარებლების უფლებებს როგორ ამოწმებს. სერვლეტების აღწერები შეგიძლიათ იგივე urls.txt ფაილში გააკეთოთ ან შექმნათ ცალკე java ფაილები. არ მოგეთხოვებათ მათი სრული იმპლემენტაცია, საკმარისია მიუთითოთ სხვადასხვა HTTP ტიპის მოთხოვნებზე როგორ იქცევა.
- თუ თქვენს იმპლემენტაციას ჭირდება Servlet context-ის ან სესიების გამოყენება, თითოეულისთვის აღწერეთ რას ინახავთ მასში და როგორ იყენებთ.

შეფასებისას ასევე ყურადღება მიექცევა ბაზასთან კავშირს რა ინტერფეისებით და მოდელის კლასებით ააწყობთ. საკმარისია ამ კლასების ინტერფეისების ახსნა, დეტალური იმპლემენტაცია არ არის საჭირო.