Applications

1. Collection et Stream

Attention : simple application JSE

Sur une liste d'entier :

- Afficher la somme
- Afficher la moyenne
- Sélectionner que les nombres qui sont > 6
- Compter le nombre d'élément

Voici une liste de personne (classe Personne à créer rapidement) :

```
public static List<Personne> getSamplePersonnes(){
        List<Personne> personnes = new ArrayList<Personne>();
        personnes.add(new Personne("Roche", "Charlotte", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Michel", "Nathan", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Lacroix MD", "Quentin",
LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Dufour", "Louis", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Dumas", "Clémence",
LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Clara", "Vincent", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Carla", "Huet", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Maëlle", "Blanchard",
LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Justine", "Brun", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Noémie", "Fournier", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Jean", "Dupond", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Pierre", "Dupont", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Jacques", "Dupou", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Thierry", "Dapont", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Jean", "Bille", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Thomas", "Artoh", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Romain", "Dupont", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Dufour", "Charlotte", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        personnes.add(new Personne("Roche", "Huet", LocalDate.of(ThreadLocalRandom.current().nextInt(1950,2021),
ThreadLocalRandom.current().nextInt(1,13),ThreadLocalRandom.current().nextInt(1,29));
        return personnes;
  }
```

Sur cette liste, avec les streams et les lamba :

- Afficher la liste en console
- Les trier par le nom puis le prénom
- Afficher que les personnes qui ont leurs noms qui commence par D
- Afficher que les noms des personnes
- Afficher le nom en majuscule et le prénom des personnes trié par ordre descendant sur le prénom dont leurs noms commence par D
- Sélectionner que les 2 première personne
- Sélection les personnes rentrés en 7ème 8ème et 9ème position
- Récupérer la personne qui a le nom Dupont
- Savoir si notre liste contient une personne dont le nom comment par D
- Regrouper les personnes par leurs prénoms
- Afficher la personne la plus jeune
- Afficher la personne la plus âgée.

2. API REST Spring

Créer une API Rest avec Spring boot “BookStore”.
Cette API gére le CRUD des objets suivants:

- Auteur : id, nom, prénom
- Livre : id, titre, année de parution, auteur
- Livre pour enfant (KidBook) qui hérite de livre avec une valeur pour donner l’âge de lecture.

Créer les 3 contrôleurs pour ces trois objets avec au minimum une route pour le getAll, le getById et le add.

Facultatif : Si vous avez le temps, ajouter les clients et les commandes.

3. Inclure JDBC

Persister les données de votre API REST en utilisant JDBC.

4. Passer à JPA

Tout en gardant le code JDBC dans votre projet, migrer la persistance de données en utilisant JPA. Nous ajoutons ici une route dans notre API pour avoir les livres écrits avant une année donnée. Pour ce faire, nous vous demandons de créer un custom repo pour les livres et d'utiliser les query avec JPQl.

5. Actuators

Inclure les endpoints pour heath et info dans votre api rest.
Créer un endpoint pour indiquer le nombre de livres dans notre application.

6. Security

Mettre en place un système d’authentification par login/pass sur notre API.

Créer l'entité User (id, username, password). Créer le repo correspondant :

```
public interface UserRepository extends JpaRepository<User, Integer>
{
      @Query(" select u from User u " + " where u.username = ?1")
      Optional<User> findUserWithName(String username);
}
```

Nous allons passer par un héritage de la classe DaoAuthenticationProvider pour effectuer l'authentification.

```
public class AppAuthProvider extends DaoAuthenticationProvider {

    @Autowired
    UserService userDetailsService;

    @Override
   public Authentication authenticate(Authentication authentication)
throws AuthenticationException {
           UsernamePasswordAuthenticationToken auth =
      (UsernamePasswordAuthenticationToken) authentication;

           String name = auth.getName();
       String password = auth.getCredentials().toString();
           UserDetails user =
      userDetailsService.loadUserByUsername(name);

           if (user == null) {
                 throw new BadCredentialsException("Username/Password
does not match for " + auth.getPrincipal());
           }

           return new UsernamePasswordAuthenticationToken(user,
null, user.getAuthorities());
    }

    @Override
    public boolean supports(Class<?> authentication) {
        return true;

    }
}
```

Créer le service user.

```
@Service
@Slf4j

public class UserService implements UserDetailsService {

    private final UserRepository userRepository;

    @Autowired
```

```
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

        Objects.requireNonNull(username);
        User user = userRepository.findUserWithName(username)
                .orElseThrow(() -> new
UsernameNotFoundException("User not found"));

        return user;
    }

}
```

Et terminer par la configuration qui indique /login comme route de connexion et /lout pour la déconnexion

7. Reactive Rest API

Développer en reactive une API REST uniquement pour les auteurs.