1.2. Les collections : TP Cyclistes, les Stream

JAVA AVANCE
(IEF Christophe Louër)

Les ***Stream*** sont arrivés avec la version 8 de Java.
Sur le TP Cycliste, implémentez une classe ***DemoStream*** dans laquelle vous alimenterez une liste de cyclistes à l'aide des méthodes suivantes que vous copierez :

```
public static Equipe creerArkeaSamsic() throws ConstructionException, PersonneException,
EquipeException {
	Equipe arkeaSamsic = EntitiesFactory.fabriquerEquipe("ARKEA_SAMSIC",
EntitiesFactory.fabriquerPersonne("Hinault", "sébastien", LocalDate.of(1965, Month.APRIL, 1)));
	arkeaSamsic.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Barguil", "Waren",
LocalDate.of(1991, Month.OCTOBER, 28),TypeSpecialite.LEADER));
	arkeaSamsic.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Greipel",
"André",LocalDate.of(1982, Month.JULY, 16),TypeSpecialite.SPRINTER));
	arkeaSamsic.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Bouet", "Maxime",
LocalDate.of(1986, Month.NOVEMBER, 3),TypeSpecialite.DOMESTIQUE));
	arkeaSamsic.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Moinard",
"Amael",LocalDate.of(1982, Month.FEBRUARY, 2),TypeSpecialite.DOMESTIQUE));
	arkeaSamsic.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Delaplace", "Anthony",
LocalDate.of(1982, Month.SEPTEMBER, 11),TypeSpecialite.DOMESTIQUE));
	arkeaSamsic.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Ledanois",
"Kevin",LocalDate.of(1993, Month.JULY, 13),TypeSpecialite.DOMESTIQUE));
	return arkeaSamsic;
}

public static Equipe creerAg2rLaMondiale() throws ConstructionException, PersonneException,
EquipeException {
	Equipe ag2r = EntitiesFactory.fabriquerEquipe("ag2r la mondiale",
EntitiesFactory.fabriquerPersonne("Hinault", "sébastien", LocalDate.of(1965, Month.APRIL, 1)));
	ag2r.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Bardet", "", LocalDate.of(1990,
Month.NOVEMBER, 9),TypeSpecialite.LEADER));
	ag2r.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Cherel", "", LocalDate.of(1986,
Month.MARCH, 17),TypeSpecialite.DOMESTIQUE));
	ag2r.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Gallopin", "", LocalDate.of(1988,
Month.MAY, 24),TypeSpecialite.BAROUDEUR));
	return ag2r;
}

public static Equipe creerQuickStep() throws ConstructionException, PersonneException,
EquipeException {
	Equipe quickstep = EntitiesFactory.fabriquerEquipe("Quick step",
EntitiesFactory.fabriquerPersonne("Hinault", "sébastien", LocalDate.of(1965, Month.APRIL, 1)));
	quickstep.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Alaphilippe", "",
LocalDate.of(1992, Month.JUNE, 11),TypeSpecialite.LEADER));
	quickstep.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Asgreen", "",
LocalDate.of(1995, Month.FEBRUARY, 8),TypeSpecialite.DOMESTIQUE));
	quickstep.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Lampaert", "",
LocalDate.of(1981, Month.APRIL, 10),TypeSpecialite.PUNCHEUR));
	quickstep.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Mørkøv", "",
LocalDate.of(1985, Month.APRIL, 30),TypeSpecialite.PUNCHEUR));
	quickstep.ajouterCycliste(EntitiesFactory.fabriquerCycliste("Viviani", "",
LocalDate.of(1989, Month.FEBRUARY, 7),TypeSpecialite.DOMESTIQUE));
	return quickstep;
}
```

Voici le squelette de méthodes que vous devez coder en utilisant les ***Stream*** :

```
/**
 * @return une liste avec les noms et prénoms des cyclistes en majuscules.
 */
public static List<String> retournerNomsPrenomsCycliste(List<Cycliste> lst) {
	// TODO : à coder
	return null;
}

/**
 * @return la liste de Cyclistes ayant une frequence cardiaque max superieure à 160.
 */
public static List<Cycliste> retournerCyclistesFrequencePlusDe160(List<Cycliste> lst) {
	// TODO : à coder
	return null;
}

/**
 * @return le cycliste qui a la frequence cardiaque maximale la + élevée.
 */
public static Cycliste retournerCyclisteFrequenceMax(List<Cycliste> lst) {
	// TODO : à coder
	return null;
}

/**
 * Calcule la frequence cardiaque maximale moyenne.
 * @return
 */
public static Double calculerFrequenceCardiaqueMaxMoyenne(List<Cycliste> lst) {
	// TODO : à coder
	return null;
}

/**
 * @return un Stream de cyclistes triés par age.
 */
public static Stream<Cycliste> retournerCyclisteParAge(List<Cycliste> lst) {
	// TODO : à coder
	return null;
}

/**
 * Affiche les 2 cyclistes les plus vieux.
 */
public static void afficherLesDeuxPlusVieuxCycliste(List<Cycliste> lst) {
	// TODO : à coder
}

/**
 * @return la liste des spécialités distinctes utilisées par les cyclistes
 */
public static List<TypeSpecialite> retournerSpecialites(List<Cycliste> lst) {
	return null;
}

/**
 * Nombre de spécialités distinctes.
 * @return
 */
```

```java
public static long compterSpecialites(List<Cycliste> lst) {
	// TODO : à coder
	return 0;
}

/**
 * @return la liste des leaders
 */
public static List<Cycliste> retournerLesLeader(List<Cycliste> lst) {
	// TODO : à coder
	return null;
}
```