Ejercicio 1

```
public Boolean existeUsuarioEnProvinciasVacunadoEnSuCumpleaños(Set<String> provincias) {
	Predicate<Vacunado> p1 = v -> v.getFechaNacimiento().getMonth()
		.equals(v.getFechaAdministracion().getMonth());
	Predicate<Vacunado> p2 = v -> v.getFechaNacimiento().getDayOfMonth()
		== v.getFechaAdministracion().getDayOfMonth();

	return vacunas.stream()
		.filter(v -> provincias.contains(v.getProvincia()))
		.anyMatch(p1.and(p2));
}
```

Solución alternativa:

```
public Boolean existeUsuarioEnProvinciasVacunadoEnSuCumpleaños(Set<String> provincias) {
	Predicate<Vacunado> p0 = v -> provincias.contains(v.getProvincia());
	Predicate<Vacunado> p1 = v -> v.getFechaNacimiento().getMonth()
		.equals(v.getFechaAdministracion().getMonth());
	Predicate<Vacunado> p2 = v -> v.getFechaNacimiento().getDayOfMonth()
		== v.getFechaAdministracion().getDayOfMonth();

	return vacunas.stream()
		.anyMatch(p0.and(p1).and(p2));
}
```

Ejercicio 2

```
public Double getEdadMediaUsuariosProvinciaPautaCompleta(String provincia) {
	return vacunas.stream()
		.filter(v -> v.getProvincia().equals(provincia) && v.getPautaCompleta())
		.mapToInt(Vacunado::getEdad)
		.average()
		.getAsDouble();
}
```

Ejercicio 3

```
public Map<String, Set<Marca>> getMarcasPorProvincia() {
	Map<String, Set<Marca>> res = new HashMap<>();

	for (Vacunado v: vacunas) {
		String provincia = v.getProvincia();
		if (res.containsKey(provincia)) {
			res.get(provincia).add(v.getMarca());
		} else {
			Set<Marca> s = new HashSet<>();
			s.add(v.getMarca());
			res.put(provincia, s);
		}
	}
	return res;
}
```

Ejercicio 4

```
public List<String> getNombresUsuariosMayorEdadMarca(Marca marca, Integer n) {
	Comparator<Vacunado> c = Comparator.comparing(Vacunado::getEdad)
		.thenComparing(Vacunado::getFechaAdministracion)
		.reversed();

	return vacunas.stream()
		.filter(v -> v.getMarca().equals(marca) && v.getPautaCompleta())
		.sorted(c)
		.limit(n)
		.map(Vacunado::getUsuario)
		.collect(Collectors.toList());
}
```

Ejercicio 5

```
public Map<String, Integer> getEdadUsuarioMasJovenPorProvincia() {
	return vacunas.stream()
		.filter(v -> v.getPautaCompleta())
		.collect(Collectors.groupingBy(
			Vacunado::getProvincia,
			Collectors.collectingAndThen(
				Collectors.minBy(
					Comparator.comparing(Vacunado::getFechaNacimiento)
						.reversed()
						.thenComparing(Comparator.naturalOrder())),
				o -> o.get().getEdad())));
}
```

Ejercicio 6

```
public Map<Marca, Double> getPorcentajeVacunadosPorMarca(Integer edadMin,
		Integer edadMax) {

	Map<Marca, Double> res = null;

	Map<Marca, Long> numeroUsuariosPorMarca = vacunas.stream()
		.filter(v -> v.getEdad() >= edadMin && v.getEdad() <= edadMax)
		.collect(Collectors.groupingBy(Vacunado::getMarca,
			Collectors.counting()));

	Long numeroUsuariosVacunados = vacunas.stream()
		.filter(v -> v.getEdad() >= edadMin && v.getEdad() <= edadMax)
		.count();

	if (numeroUsuariosVacunados > 0) {
	    res = numeroUsuariosPorMarca.entrySet().stream()
		.collect(Collectors.toMap(
			e -> e.getKey(),
			e -> e.getValue() * 100.0 / numeroUsuariosVacunados));
	}

	return res;
}
```

Solución alternativa:

```
public Map<Marca, Double> getPorcentajeVacunadosPorMarca(Integer edadMin,
		Integer edadMax) {

	Map<Marca, Double> res = null;

	Long numeroUsuariosVacunados = vacunas.stream()
		.filter(v -> v.getEdad() >= edadMin && v.getEdad() <= edadMax)
		.count();

	if (numeroUsuariosVacunados > 0) {
		res = vacunas.stream()
			.collect(Collectors.groupingBy(Vacunado::getMarca,
				Collectors.collectingAndThen(
					Collectors.counting(),
					c -> c * 100.0 / numeroUsuariosVacunados)));
	}

	return res;
}
```

Ejercicio 7

```
public LocalDate getFechaMasDosisMarcaProvincia(Marca marca, String provincia) {

	Map<LocalDate, Long> numeroVacunasPorFecha = vacunas.stream()
		.filter(v -> v.getMarca().equals(marca)
			&& v.getProvincia().equals(provincia))
		.collect(Collectors.groupingBy(
			Vacunado::getFechaAdministracion,
			Collectors.counting()));

	return numeroVacunasPorFecha.entrySet().stream()
		.max(Comparator.comparing(e -> e.getValue()));
		.get().getKey();
}
```