FUNDAMENTOS DE PROGRAMACIÓN. Curso 2023/24

SEGUNDO PARCIAL. 28 de mayo de 2024. Soluciones

Ejercicio 1

```
public record Resultado(String partido, Double porcentaje) {
	public Resultado {
		Checkers.check("Resultado::El porcentaje no puede ser negativo",
		porcentaje >= 0);
	}
}
```

Ejercicio 2

```
public class Encuesta implements Comparable<Encuesta> {
	private String consultora;
	private LocalDate fechaComienzo;
	private LocalDate fechaFin;
	private Integer numeroEncuestados;
	private String pais;
	private TipoEncuesta tipo;
	private Double porcentajeIndecisos;
	private List<Resultado> resultados;

	public Encuesta(String consultora, LocalDate fechaComienzo,
			LocalDate fechaFin, Integer numeroEncuestados, String pais,
			TipoEncuesta tipo, Double porcentajeIndecisos,
			List<Resultado> resultados) {

		Checkers.check("Encuesta::FechaComienzo debe ser anterior a fecha
		fin", fechaComienzo.isBefore(fechaFin));

		Checkers.check("Encuesta::La lista de resultados no puede estar
		vacía", !resultados.isEmpty());

		Checkers.check("Encuesta::El número de encuestados debe ser mayor o
		igual que cero", numeroEncuestados >= 0);

		this.consultora = consultora;
		this.fechaComienzo = fechaComienzo;
		this.fechaFin = fechaFin;
		setNumeroEncuestados(numeroEncuestados);
		this.pais = pais;
		this.tipo = tipo;
		this.porcentajeIndecisos = porcentajeIndecisos;
		this.resultados = new ArrayList<>(resultados);
	}

	public Integer getNumeroEncuestados() {
		return numeroEncuestados;
	}

	public void setNumeroEncuestados(Integer numeroEncuestados) {
		Checkers.check("Encuesta::El número de encuestados debe ser mayor o
		igual que cero", numeroEncuestados >= 0);
		this.numeroEncuestados = numeroEncuestados;
	}

	public Double getPorcentajeIndecisos() {
		return porcentajeIndecisos;
	}
```

```java
public void setPorcentajeIndecisos(Double porcentajeIndecisos) {
	this.porcentajeIndecisos = porcentajeIndecisos;
}

public List<Resultado> getResultados() {
	return new ArrayList<Resultado>(resultados);
}

public String getConsultora() {
	return consultora;
}

public LocalDate getFechaComienzo() {
	return fechaComienzo;
}

public LocalDate getFechaFin() {
	return fechaFin;
}

public String getPais() {
	return pais;
}

public TipoEncuesta getTipo() {
	return tipo;
}

public Double getRatioEncuestadosDia() {
	return numeroEncuestados * 1.0 / (Period.between(fechaComienzo,
		fechaFin).getDays());
	// También se puede usar en lugar de Period.betweem
	// fechaComienzo.until(fechaFin).getDays()
}

public String toString() {
	return "Encuesta [consultora=" + consultora + ", fechaComienzo=" +
	fechaComienzo + ", fechaFin=" + fechaFin
	+ ", numeroEncuestados=" + numeroEncuestados + ", pais=" + pais + ",
	tipo=" + tipo + ", porcentajeIndecisos=" + porcentajeIndecisos + ",
	resultados=" + resultados + "]";
}

public int hashCode() {
	return Objects.hash(fechaComienzo, fechaFin, consultora,
	numeroEncuestados);
}

public boolean equals(Object obj) {
	if (this == obj)
		return true;
	if (!(obj instanceof Encuesta))
		return false;
	Encuesta other = (Encuesta) obj;
	return Objects.equals(fechaComienzo, other.fechaComienzo) &&
	Objects.equals(fechaFin, other.fechaFin)&& Objects.equals(consultora,
	other.consultora) && Objects.equals(numeroEncuestados,
	other.numeroEncuestados);
}
```

```
	public int compareTo(Encuesta e) {
		int res= getFechaComienzo().compareTo(e.getFechaComienzo());

		if (res == 0) {
			res = getFechaFin().compareTo(e.getFechaFin());
			if (res == 0) {
				res = getConsultora().compareTo(e.getConsultora());
				if (res == 0) {
					res = getNumeroEncuestados().compareTo(
						e.getNumeroEncuestados());
				}
			}
		}
		return res;
	}
}
```

Ejercicio 3

```
private static Encuesta parsearEncuesta (String lineaCSV) {
	Checkers.checkNoNull(lineaCSV);
	String [] trozos = lineaCSV.split(",");
	Checkers.check("FactoriaEncuestas::Formato no válido", trozos.length == 8);
	String consultora = trozos[0].strip();
	LocalDate fechaInicio = LocalDate.parse(trozos[1].strip());
	LocalDate fechaFin = LocalDate.parse(trozos[2].strip());
	Integer numEncuestados = Integer.valueOf(trozos[3].strip());
	String pais = trozos[4].strip();
	TipoEncuesta tipo = parseaTipoEncuesta(trozos[5].strip());
	Double porcentajeIndecisos = Double.valueOf(trozos[6].strip());
	List<Resultado> resultados = parseaResultados(trozos[7].strip());

	return new Encuesta(consultora, fechaInicio, fechaFin, numEncuestados,
		pais, tipo, porcentajeIndecisos, resultados);
}

private static List<Resultado> parseaResultados(String cad) {
	String clean = cad.replace ("'", "")
			  .replace("(", "")
			  .replace(")", "");
	String [] trozos = clean.split(";");
	List<Resultado> res = new ArrayList<>();
	for (String trozo:trozos) {
		res.add(parseaResultado(trozo.strip()));
	}
	return res;
}

private static Resultado parseaResultado(String trozo) {
	String [] trozos = trozo.split(":");
	Checkers.check("Formato resultado no válido", trozos.length == 2);
	Double porcentaje = Double.valueOf(trozos[1].strip());
	return new Resultado(trozos[0].strip(), porcentaje);
}

private static TipoEncuesta parseaTipoEncuesta(String cad) {
	return TipoEncuesta.valueOf(cad.toUpperCase());
}
```

Ejercicio 4

Apartado 4.1

```
public Double getMediaNumEncuestadosConsultorayFecha(String consultora,
            LocalDate fechaMaxima) {
      return encuestas.stream()
            .filter(e -> e.getConsultora().equals(consultora) &&
                  e.getFechaFin().isBefore(fechaMaxima))
            .mapToInt(Encuesta::getNumeroEncuestados)
            .average()
            .orElse(0.);
}
```

Apartado 4.2

```
public Encuesta getEncuestaMasEncuestadosPorDia(TipoEncuesta tipo) {
      Comparator<Encuesta> c =
            Comparator.comparing(Encuesta::getRatioEncuestadosDia);
      return encuestas.stream()
            .filter(e -> e.getTipo().equals(tipo))
            .max(c)
            .get();
}
```

Apartado 4.3

```
public List<String> getPartidosMasFrecuentesOrdenados(Integer n) {
      Map<String, Long> contarPartidos =
            encuestas.stream().flatMap(e -> e.getResultados().stream())
            .collect(Collectors.groupingBy(
                  Resultado::partido, Collectors.counting()));
      Comparator<Map.Entry<String, Long>> c = Map.Entry.comparingByValue();

      return contarPartidos.entrySet().stream()
            .sorted(c.reversed())
            .limit(n)
            .map(Map.Entry::getKey)
            .collect(Collectors.toList());
}
```

Apartado 4.4

```
public SortedMap<String, Boolean> getSuperaEncuestadosPorPais(Integer umbral) {
      return encuestas.stream()
            .collect(Collectors.groupingBy(
                  Encuesta::getPais,
                  TreeMap::new,
                  Collectors.collectingAndThen(
                        Collectors.toList(),
                        encuestas -> todasEncuestasSuperanUmbral(
                              encuestas, umbral))));
}

private Boolean todasEncuestasSuperanUmbral(List<Encuesta> encuestas,
            Integer umbral) {
      return encuestas.stream()
            .allMatch(e -> e.getNumeroEncuestados() > umbral);
}
```

Apartado 4.5

```
public Map<String, SortedSet<String>> getPaisesPorPartidoMayorPorcentaje(
            Double umbralPorcentaje) {
      Map<String, SortedSet<String>> res = new HashMap<>();
      for (Encuesta e: encuestas) {
            for (Resultado r: e.getResultados()) {
                  if (r.porcentaje()>umbralPorcentaje) {
                        String clave = r.partido();
                        if (res.containsKey(clave)) {
                              res.get(clave).add(e.getPais());
                        } else {
                              SortedSet<String> c = new TreeSet<>();
                              c.add(e.getPais());
                              res.put(clave, c);
                        }
                  }
            }
      }
      return res;
}
```

Test

```
public class TestEstadisticasEncuestas {

      public static void main(String[] args) {
            EstadisticasEncuestas est = FactoriaEncuestas.creaEstadisticas(
                  "data/encuestas_electorales.csv");
            testCreaEstadisticas(est);
            System.out.println("EJ1" + " =".repeat(80));
            testGetMediaNumEncuestadosConsultorayFecha(est, "Consultora A",
                  LocalDate.of(2024, 5, 29));
            testGetMediaNumEncuestadosConsultorayFecha(est, "Consultora B",
                  LocalDate.of(2024, 5, 26));
            System.out.println("\nEJ2" + " =".repeat(80));
            testGetResultadosMasEncuestadosPorDia(est, TipoEncuesta.TELEFONICA);
            testGetResultadosMasEncuestadosPorDia(est, TipoEncuesta.PRESENCIAL);
            System.out.println("\nEJ3" + " =".repeat(80));
            testGetPartidosMasFrecuentesOrdenados(est, 5);
            System.out.println("\nEJ4" + " =".repeat(80));
            testGetSuperaEncuestadosPorPais(est, 3500);
            System.out.println("\nEJ5" + " =".repeat(80));
            testGetPaisesPorPartidoMayorPorcentaje(est,90.);
            testGetPaisesPorPartidoMayorPorcentaje(est,97.);
      }

      private static void testCreaEstadisticas(EstadisticasEncuestas est) {
            try {
                  List<Encuesta> encuestas = est.getEncuestas();
                  System.out.println(String.format("Leidas %d encuestas",
                        encuestas.size()));
                  int i = 0;
                  for (Encuesta e: encuestas) {
                        System.out.println("\t" + (i++) + "-" + e);
                  }
            } catch (Exception e) {
                  System.out.println(e.getMessage());
            }
      }
```

```
	private static void testGetMediaNumEncuestadosConsultorayFecha(
			EstadisticasEncuestas est, String consultora, LocalDate fechaMaxima){
		try {
			Double m = est.getMediaNumEncuestadosConsultorayFecha(
				consultora, fechaMaxima);
			String msg = String.format("La media de la consultora %s con
			fecha de fin anterior a %s es %f", consultora, fechaMaxima, m);
			System.out.println(msg);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}

	private static void testGetResultadosMasEncuestadosPorDia(
			EstadisticasEncuestas est, TipoEncuesta tipo) {
		try {
			Encuesta r = est.getEncuestaMasEncuestadosPorDia(tipo);
			String msg = String.format("\tLa encuesta de tipo %s con mayor
			ratio de encuestados por día es :\n%s ", tipo, r);
			System.out.println(msg);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}

	private static void testGetPartidosMasFrecuentesOrdenados(
			EstadisticasEncuestas est, Integer n) {
		try {
			List<String> r = est.getPartidosMasFrecuentesOrdenados(n);
			String msg = String.format("\tLos %d partidos más frecuentes en
			las encuestas son:\n%s", n, r);
			System.out.println(msg);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}

	private static void testGetSuperaEncuestadosPorPais(
			EstadisticasEncuestas est, Integer umbral) {
		try {
			SortedMap<String, Boolean> r =
				est.getSuperaEncuestadosPorPais(umbral);
			String msg = String.format("\t¿Todas las encuestas de estos
			países superan los %s encuestados?:\n%s ", umbral, r);
			System.out.println(msg);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}

	private static void testGetPaisesPorPartidoMayorPorcentaje(
			EstadisticasEncuestas est, Double umbral) {
		try {
			Map<String, SortedSet<String>> r =
				est.getPaisesPorPartidoMayorPorcentaje(umbral);
			String msg = String.format("\tPaíses donde cada partido ha
			obtenido un porcentaje de votos superior a %1f :\n%s ", umbral, r);
			System.out.println(msg);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```