	FUNDAMENTOS DE PROGRAMACIÓN **EXAMEN DE TEORÍA. BLOQUE 1 DE JAVA. SOLUCIONES**	**Curso: 2024/25**
APELLIDOS:	**NOMBRE:**	**DNI:**

Sean las siguientes definiciones de tipos:

Tipo **Ruta**
Propiedades:

- nombre: String, consultable
- longitud: Double, consultable (distancia en kilómetros)
- circular: Boolean, consultable. Toma valor true si la ruta es circular y false si es lineal
- tiempo: Integer, consultable (tiempo estimado en horas)
- dificultad: del tipo enumerado DificultadRuta (FACIL, MODERADA, DIFICIL), consultable

Tipo **Excursión**
Propiedades:

- fecha: LocalDate, consultable y modificable
- horaSalida: LocalTime, consultable y modificable
- horaLlegada: LocalTime, consultable
- ruta: Ruta, consultable y modificable
- numeroParticipantes: Integer, consultable y modificable
- guia: Boolean, consultable. Toma valor true si la excursión requiere guía y false si no. Una excursión requiere guía si tiene más de 20 participantes, si la ruta es DIFICIL o si es MODERADA y tiene una longitud superior a 10 kilómetros.

El tipo Ruta está implementado mediante un **record** y el tipo Excursión mediante una **clase**. Suponga que los métodos consultores y modificadores de la clase se nombran anteponiendo la palabra get o set, respectivamente, al nombre de la propiedad. Por ejemplo, getFecha y setFecha.

1. Dados un objeto r de tipo Ruta y un objeto e de tipo Excursión, escriba una expresión lógica que tome valor true si se cumple cada una de las siguientes condiciones, y false en caso contrario: (2 puntos)
   a) El nombre de la ruta r tiene 5 caracteres y el primero es una letra
   b) La excursión e tiene más de 20 participantes y comienza antes de las 9 de la mañana
   c) La fecha de la excursión e es igual o posterior a la de hoy
   d) La ruta r es circular y tiene una dificultad MODERADA
   e) El nombre de la ruta r comienza por las letras “PR”
   f) La excursión e se realiza un sábado
   g) El nombre de la ruta r no está vacío
   h) La ruta r es fácil y, o bien tiene menos de 10 km, o es circular

```
a) r.nombre().length() == 5 && Character.isLetter(r.nombre().charAt(0))
b) e.getNumeroParticipantes() > 20 && e.getHoraSalida().getHour() < 9
c) !e.getFecha().isBefore(LocalDate.now())
d) r.circular() && r.dificultad().equals(DificultadRuta.MODERADA)
e) r.nombre().startsWith("PR")
f) e.getFecha().getDayOfWeek().equals(DayOfWeek.SATURDAY)
g) !r.nombre().isEmpty()
h) r.dificultad().equals(DificultadRuta.FACIL) && (r.longitud() < 10 ||
   r.circular())
```

2. Dado un objeto e de tipo Excursión, escriba las instrucciones que realizan lo que se pide en cada caso (si algo de lo que se pide no es posible, indíquelo y justifíquelo): (1 punto)
   a) Modificar la fecha de la excursión e por la fecha 21/3/2025
   b) Cambiar a MODERADA la dificultad de la ruta de la excursión e
   c) Retrasar una hora la hora de salida de la excursión e
   d) Crear una cadena formada por el nombre de la ruta y el número de participantes de la excursión e, en el formato dado en el siguiente ejemplo: "Sendero de las laderas – 30 asistentes"

```
a) e.setFecha(LocalDate.of(2025, 3, 21));
b) No es posible cambiar el valor de un atributo de un record porque es un tipo
   inmutable
c) e.setHoraSalida(e.getHoraSalida().plusHours(1));
d) String cadenaRuta = e.getRuta().nombre() + " - " +  e.getNumeroParticipantes() + "
   asistentes";
```

3. Escriba el código del método getGuia de la clase Excursión. (1 punto)

```
public Boolean getGuia() {
	Boolean res = false;
	if (numeroParticipantes > 20
			|| ruta.dificultad().equals(DificultadRuta.DIFICIL)
			|| (ruta.dificultad().equals(DificultadRuta.MODERADA) &&
				ruta.longitud() > 10)) {
		res = true;
	}
	return res;
}
```

4. Escriba el código del método getHoraLlegada de la clase Excursión. La hora de llegada se obtiene sumando a la hora de salida el tiempo estimado de la ruta. (0,5 puntos)

```
public LocalTime getHoraLlegada() {
	return horaSalida.plusHours(getRuta().tiempo());
}
```

5. Las excursiones se ordenan por su fecha, y a igualdad de esta por su hora de salida. Escriba todo lo que debe incluir en la clase para que el tipo Excursión sea comparable y tenga este orden natural. (1 punto)

```
public class Excursion implements Comparable<Excursion> {
...
	public int compareTo(Excursion o) {
		int res = fecha.compareTo(o.getFecha());
		if (res == 0) {
			res = horaSalida.compareTo(o.getHoraSalida());
		}
		return res;
	}
```

	FUNDAMENTOS DE PROGRAMACIÓN EXAMEN DE TEORÍA. BLOQUE 1 DE JAVA. SOLUCIONES	Curso: 2024/25
APELLIDOS:	NOMBRE:	DNI:

6. Escriba un método getTipoDia en la clase Excursion que devuelva una cadena con el valor “Día laborable” si el día de la semana en que se realiza la excursión es de lunes a viernes, y “Fin de semana” si es un sábado o domingo. Utilice la estructura switch y el método getDayOfWeek del tipo LocalDate, que devuelve un enumerado de tipo DayOfWeek que toma los valores MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY y SUNDAY. (1 punto)

```
public String getTipoDia() {
	String res = null;
	switch (fecha.getDayOfWeek()) {
	case MONDAY: case TUESDAY: case WEDNESDAY: case THURSDAY: case FRIDAY:
		res = "Día laborable";
		break;
	case SATURDAY: case SUNDAY:
		res = "Fin de semana";
	}
	return res;
}
```

7. Escriba las instrucciones necesarias para realizar las siguientes operaciones: (1,5 puntos)
   a) Crear una lista de excursiones de nombre excursiones2025 a partir de la lista existente excursiones
   b) Añadir al final de la lista excursiones2025 la excursión e1
   c) Añadir en la primera posición de la lista excursiones2025 la excursión e2
   d) Añadir a la lista excursiones2025 todas las excursiones del conjunto otrasExcursiones
   e) Eliminar de la lista excursiones2025 la excursión e1
   f) Ordenar la lista excursiones2025 por el orden natural del tipo Excursion

```
a) List<Excursion> excursiones2025 = new ArrayList<>(excursiones);
b) excursiones2025.add(e1);
c) excursiones2025.add(0, e2);
d) excursiones2025.addAll(otrasExcursiones);
e) excursiones2025.remove(e1);
f) Collections.sort(excursiones2025);
```

8. Sea excursiones una lista de excursiones. Escriba un trozo de código que cree una lista con los nombres de las rutas de más de 5 horas realizadas en el año 2025. (1 punto)

```
List<String> res = new ArrayList<>();
for (Excursion e: excursiones) {
	if (e.getRuta().tiempo() > 5 && e.getFecha().getYear() == 2025) {
		res.add(e.getRuta().nombre());
	}
}
```

9. Sea excursiones una lista de excursiones. Escriba un trozo de código que obtenga un valor lógico true si existe alguna excursión que no requiera guía y tenga una ruta circular, y false en caso contrario. (1 punto)

```
Boolean existe = false;
for (Excursion e: excursiones) {
	if (e.getRuta().circular() && !e.getGuia()) {
		existe = true;
		break;
	}
}
```