Parte de nuestra aplicación ha de crear instancias de la clase Email, que son inmutables, y contiene algunos campos obligatorios y otros optativos (para simplificar el enunciado, hemos supuesto que son todos de tipo String). Un esbozo de la misma es:

```
public final class Email {
  private final String from;
  private final List<String> to;
  private final String subject;
  private final String body;
  private final List<String> ccTo;

  public Email(
      String from, List<String> to, String subject, String body) {

    this(from, to, subject, body, new ArrayList<>());
  }

  public Email(
      String from, List<String> to, String subject, String body,
      List<String> ccTo) {

    this.from = from; this.to = to; this.subject = subject;
    this.body = body; this.ccTo = ccTo;
  }

  // more methods
}
```

Se quiere una API más cómoda para generar los correos de manera que podamos ir acumulando sus parámetros y, antes de construirlo, comprobar que tenemos toda la información y ésta es correcta.

Además, la clase Email actual es frágil, ya que no comprueba que los Emails estén bien formado, por lo que algunos métodos acaban fallando muy lejos de la creación de las instancias erróneas.

El objetivo de la nueva API será poder hacer:

```
Email email = Email.builder()
                   .from("juanmanuel.gimeno@udl.cat")
                   .to("xyz@alumnes.udl.cat")
                   .to("pqr@alumnes.udl.cat")
                   .subject("Copias en los exámenes presenciales")
                   .body("<strong>Ya sabíais las consecuencias " +
                         "de copiar ...</strong>")
                   .ccTo("capest.inf@eps.udl.cat")
                   .make();
```

De manera que los nuevos métodos de creación:

- podrán llamarse en el orden que se quiera (excepto `make`, que será el último, y `builder`, que será el primero)
- `from`, `subject`, `body` se han de llamar exactamente una vez
- `to` se ha llamar por lo menos una vez y puede llamarse varias
- `ccTo` es optativo y puede llamarse varias veces

En caso de no respetarse estas reglas, se lanzará la excepción no comprobada (unchecked exception) `EmailBuilderException`.

```
public class EmailBuilderException extends RuntimeException {
    public EmailBuilderException(String message) {
        super(message);
    }
}
```

Si se producen varios errores, solamente hace falta que la excepción que se lance indique uno de ellos.

Lo que se pide es que:

1. (2 puntos) Indiquéis qué patrón se debe aplicar y **por qué** sirve para resolver el problema planteado
2. (8 puntos) Implementéis la nueva API para crear instancias de Email. Además del código, añadid una pequeña descripción de las decisiones de implementación/diseño que tomáis

Obviamente se valorará tanto la corrección del código como su limpieza, sencillez y claridad.

NOTA: No hace falta que comprobéis que las direcciones de email están bien formadas (así nos ahorramos accesos innecesarios a https://stackoverflow.com/)

La entrega consistirá en un único PDF con vuestra solución.