# SIMULADORES

## *Using Operators and Decision Contructors*

*1.*

Test Overview Time Left - 00:09:00

| | | | |
|---|---|---|---|
| **Name** | Taken on - 26 jul, '24 12:28 AM | **Status** | Passed 88% |
| **Correct Answers** | 14 | **Total Questions** | 16 |
| **Time Taken** | 00:25:08 | **Total Time** | 00:34:08 |
| **Start Time** | 26 jul 24 00:28 | **Finish/Pause Time** | 26 jul 24 00:54 |

Test Details | Performance Report

| S ... | Marked | Atte... | Result | Exam Objective | Difficulty Le... | Problem Statement | Note |
|---|---|---|---|---|---|---|---|
| 1 | | ✓ | ✓ | 03 - Using Operator... | Real Brainer | `Object t = new Integer(107);` | |
| 2 | | ✓ | ✗ | 03 - Using Operator... | Real Brainer | | |
| 3 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | 3. | |
| 4 | | ✓ | ✗ | 03 - Using Operator... | Very Easy | `boolean flag = true;` | |
| 5 | | ✓ | ✓ | 03 - Using Operator... | Tough | `case 2:` | |
| 6 | | ✓ | ✓ | 03 - Using Operator... | Real Brainer | `System.out.println(true + null); //2` | |
| 7 | | ✓ | ✓ | 03 - Using Operator... | Easy | | |
| 8 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | `Object obj1 = new Object();` `Object obj2 = obj1;` | |
| 9 | | ✓ | ✓ | 03 - Using Operator... | Easy | `default : System.out.println(` | |
| 10 | | ✓ | ✓ | 03 - Using Operator... | Easy | `case true :` | |
| 11 | | ✓ | ✓ | 03 - Using Operator... | Tough | `System.out.println("Hello!");` | |
| 12 | | ✓ | ✓ | 03 - Using Operator... | Easy | Which of the following statements are true? | |
| 13 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | `int j = 1;` `if( (i++ == 0) & (i++ == 2) ){` | |
| 14 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | `while(checkit(k)){` `float f = 5.0f;` | |
| 15 | | ✓ | ✓ | 03 - Using Operator... | Easy | `System.out.print(k);` `double d = 3.8;` | |
| 16 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | `int a = 1;` | |

*2.*

Test Overview Time Left - 00:24:33

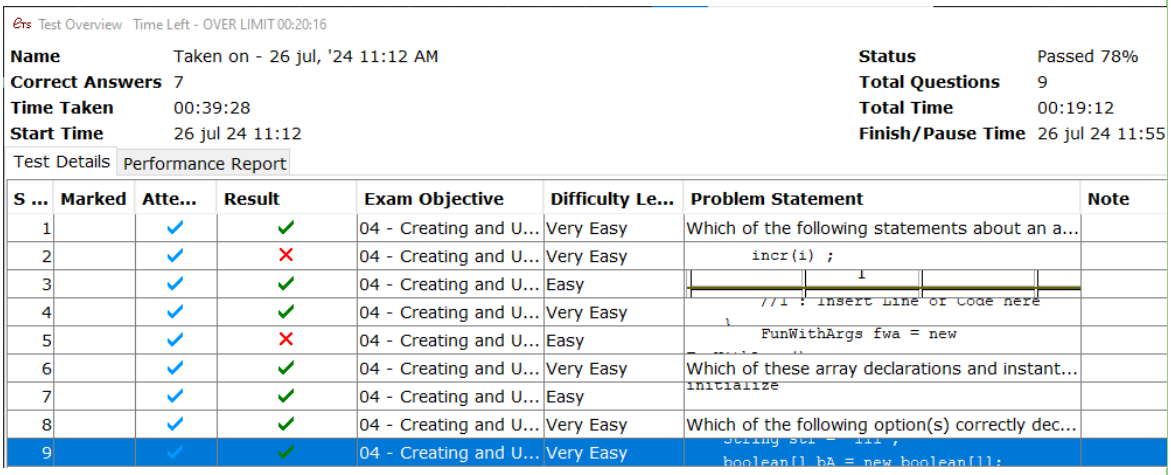
| | | | |
|---|---|---|---|
| **Name** | Taken on - 26 jul, '24 12:55 AM | **Status** | Passed 80% |
| **Correct Answers** | 16 | **Total Questions** | 20 |
| **Time Taken** | 00:18:07 | **Total Time** | 00:42:40 |
| **Start Time** | 26 jul 24 00:55 | **Finish/Pause Time** | 26 jul 24 01:13 |

Test Details | Performance Report

| S ... | Marked | Atte... | Result | Exam Objective | Difficulty Le... | Problem Statement | Note |
|---|---|---|---|---|---|---|---|
| 1 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | `while(checkit(k)){` `System.out.print(k);` | |
| 2 | | ✓ | ✗ | 03 - Using Operator... | Very Easy | `case 1:` | |
| 3 | | ✓ | ✓ | 03 - Using Operator... | Tough | `static int x = 5;` | |
| 4 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | `int a = 1;` | |
| 5 | | ✓ | ✓ | 03 - Using Operator... | Easy | `case true :` | |
| 6 | | ✓ | ✓ | 03 - Using Operator... | Easy | `int i;` | |
| 7 | | ✓ | ✓ | 03 - Using Operator... | Real Brainer | `else //4` | |
| 8 | | ✓ | ✓ | 03 - Using Operator... | Real Brainer | `System.out.println(true + null); //2` | |
| 9 | | ✓ | ✗ | 03 - Using Operator... | Real Brainer | `int[] a = { 1 };` | |
| 10 | | ✓ | ✓ | 03 - Using Operator... | Tough | `System.out.println("Hello!");` | |
| 11 | | ✓ | ✓ | 03 - Using Operator... | Easy | `throws Exception {` | |
| 12 | | ✓ | ✓ | 03 - Using Operator... | Easy | | |
| 13 | | ✓ | ✗ | 03 - Using Operator... | Real Brainer | | |
| 14 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | 3. | |
| 15 | | ✓ | ✓ | 03 - Using Operator... | Very Tough | `}` `switch(x){` | |
| 16 | | ✓ | ✗ | 03 - Using Operator... | Very Tough | `case 'b': // 1` | |
| 17 | | ✓ | ✓ | 03 - Using Operator... | Very Easy | Which of the following implementations of a `max()` method will correctly return the largest | |
| 18 | | ✓ | ✓ | 03 - Using Operator... | Tough | `}` | |
| 19 | | ✓ | ✓ | 03 - Using Operator... | Tough | `case 2:` | |
| 20 | | ✓ | ✓ | 03 - Using Operator... | Real Brainer | `String str1 = "one";` | |

### *Creating and Using Arrays*

*1.*



Test Overview  Time Left - OVER LIMIT 00:20:16

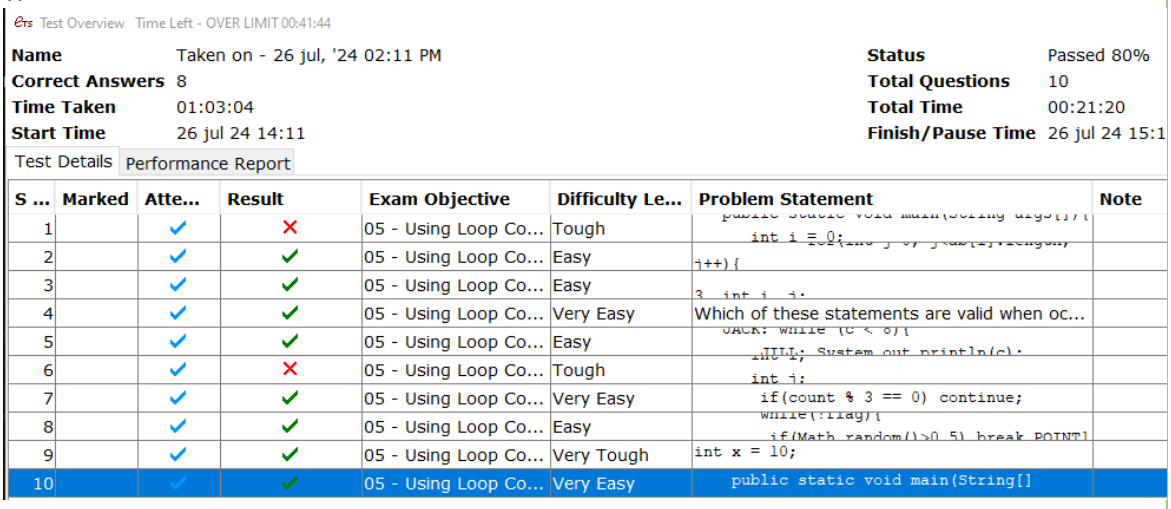
| | | | |
|---|---|---|---|
| **Name** | Taken on - 26 jul, '24 11:12 AM | **Status** | Passed 78% |
| **Correct Answers** | 7 | **Total Questions** | 9 |
| **Time Taken** | 00:39:28 | **Total Time** | 00:19:12 |
| **Start Time** | 26 jul 24 11:12 | **Finish/Pause Time** | 26 jul 24 11:55 |

Test Details | Performance Report

| S ... | Marked | Atte... | Result | Exam Objective | Difficulty Le... | Problem Statement | Note |
|---|---|---|---|---|---|---|---|
| 1 | | ✓ | ✓ | 04 - Creating and U... | Very Easy | Which of the following statements about an a... | |
| 2 | | ✓ | ✗ | 04 - Creating and U... | Very Easy | incr(i) ; | |
| 3 | | ✓ | ✓ | 04 - Creating and U... | Easy | | |
| 4 | | ✓ | ✓ | 04 - Creating and U... | Very Easy | | |
| 5 | | ✓ | ✗ | 04 - Creating and U... | Easy | FunWithArgs fwa = new | |
| 6 | | ✓ | ✓ | 04 - Creating and U... | Very Easy | Which of these array declarations and instant... | |
| 7 | | ✓ | ✓ | 04 - Creating and U... | Easy | | |
| 8 | | ✓ | ✓ | 04 - Creating and U... | Very Easy | Which of the following option(s) correctly dec... | |
| 9 | | ✓ | ✓ | 04 - Creating and U... | Very Easy | boolean[] bA = new boolean[1]; | |

### *Using Loop Constructors*

*1.*



Test Overview  Time Left - OVER LIMIT 00:41:44

| | | | |
|---|---|---|---|
| **Name** | Taken on - 26 jul, '24 02:11 PM | **Status** | Passed 80% |
| **Correct Answers** | 8 | **Total Questions** | 10 |
| **Time Taken** | 01:03:04 | **Total Time** | 00:21:20 |
| **Start Time** | 26 jul 24 14:11 | **Finish/Pause Time** | 26 jul 24 15:1 |

Test Details | Performance Report

| S ... | Marked | Atte... | Result | Exam Objective | Difficulty Le... | Problem Statement | Note |
|---|---|---|---|---|---|---|---|
| 1 | | ✓ | ✗ | 05 - Using Loop Co... | Tough | int i = 0; | |
| 2 | | ✓ | ✓ | 05 - Using Loop Co... | Easy | | |
| 3 | | ✓ | ✓ | 05 - Using Loop Co... | Easy | | |
| 4 | | ✓ | ✓ | 05 - Using Loop Co... | Very Easy | Which of these statements are valid when oc... | |
| 5 | | ✓ | ✓ | 05 - Using Loop Co... | Easy | | |
| 6 | | ✓ | ✗ | 05 - Using Loop Co... | Tough | int j; | |
| 7 | | ✓ | ✓ | 05 - Using Loop Co... | Very Easy | if(count % 3 == 0) continue; | |
| 8 | | ✓ | ✓ | 05 - Using Loop Co... | Easy | | |
| 9 | | ✓ | ✓ | 05 - Using Loop Co... | Very Tough | int x = 10; | |
| 10 | | ✓ | ✓ | 05 - Using Loop Co... | Very Easy | public static void main(String[] | |

# SPRING BATCH

Introducción

Batch hace referencia a lotes por lo que el procesamiento por lotes consiste en aquellos programas que se lanzan generalmente de manera programada y que no requieren ningún tipo de intervención humana. Los cuales se caracterizan por ser procesos relativamente pesados, que tratan una gran cantidad de información, lo que hace que se ejecuten en horario con baja carga de trabajo para no influir en el entorno transaccional.

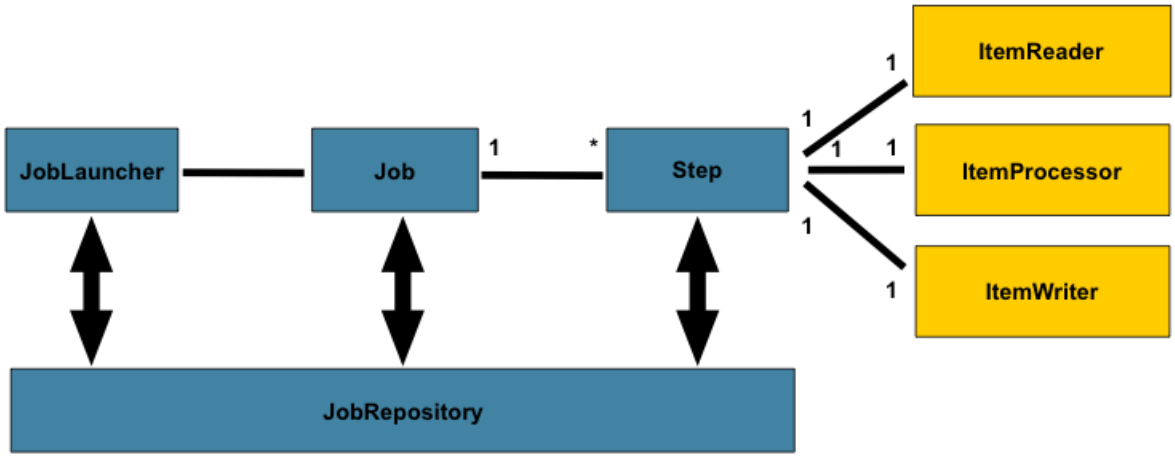
## VISIÓN GENERAL

1. ¿Qué es Spring Batch?

   Spring Batch es un framework ligero enfocado específicamente en la creación de procesos Batch. Provee funciones reutilizables que son esenciales en el procesamiento de grandes volúmenes de registros incluyendo logging/tracing, la gestión de transacciones, las estadísticas de procesamiento de trabajo, el reinicio, la omisión y la gestión de recursos. También proporciona funciones y servicios técnicos mas avanzados que permitirán trabajo por lotes de alto volumen y rendimiento a través de técnicas de optimización y partición.

2. Casos de uso y aplicaciones prácticas.
   Un ejemplo de uso es la carga de un fichero enorme con millones de registros; o bien u proceso nocturno que, a partir de una serie de consultas, envía una gran cantidad de e-mails, sms, etc.

3. Arquitectura de Spring Batch.
   Componentes de Spring Batch




En el diseño mostrado en la figura podemos observar diferentes elementos:

- **JobRepository:** es el componente encargado de la persistencia de metadatos relativos a los procesos tales como procesos en curso o estados de las ejecuciones.
- **JobLauncher:** es el componente encargado de lanzar los procesos suministrando los parámetros de entrada deseados.
- **Job**: El Job es la representación del proceso. Un proceso, a su vez, es un contenedor de pasos (steps).
- **Step:** Un step (paso) es un elemento independiente dentro de un Job (un proceso) que representa una de las fases de las que está compuesto dicho proceso. Un proceso (Job) debe tener, al menos, un step.
  Un step puede estar copuesto de tres elementos: reader,writer y processor donde:

- ItemReader: Elemento responsable de leer datos de una fuente de datos (BBDD, fichero, cola de mensajes, etc...)
- ItemProcessor: Elemento responsable tratar la información obtenida por el reader. No es obligatorio su uso.
- ItemWriter: Elemento responsable guardar la información leída por el reader o tratada por el processor. Si hay un reader debe haber un writer.

## CONFIGURACIÓN DEL ENTORNO

1. Configuración del proyecto
2. Creación de un proyecto Spring Batch con Spring Boot.
3. Dependencias necesarias (Maven/Gradle).
4. Configuración básica y avanzada.

## CONCEPTOS BÁSICOS

### DOMINIO DE SPRING BATCH

1. Introducción a Jobs y Steps.
2. Tasklets y Chunks: Diferencias y cuándo usarlos.
3. JobRepository y su configuración.

### FLUJO DE UN JOB

1. JobLauncher y JobExecution.
2. Configuración de Job y Step.
3. Listeners y eventos en Spring Batch.

## PROCESAMIENTO DE DATOS

Lectura de Datos

ItemReader y sus implementaciones (FlatFileItemReader, JdbcCursorItemReader, JpaPagingItemReader, etc.).

Configuración de múltiples fuentes de datos.

Manejo de excepciones durante la lectura.

Procesamiento de Datos

ItemProcessor y sus implementaciones.

Validación y transformación de datos.

Procesadores compuestos y cadenas de procesamiento.

Escritura de Datos

ItemWriter y sus implementaciones (FlatFileItemWriter, JdbcBatchItemWriter, JpaItemWriter, etc.).

Configuración de múltiples destinos de datos.

Manejo de transacciones y commit.

## CONTROL DE FLUJO

Control de Flujo en Jobs

Configuración de flujo condicional.

JobExecutionDecider.

Divisiones y particiones.

Escalabilidad y Paralelismo

Partitioner y Multi-threaded Step.

Remote Chunking y Remote Partitioning.

Configuración y mejores prácticas para la escalabilidad.

## MANEJO DE ERRORES Y TRANSACCIONES

Manejo de Errores

RetryTemplate y configuración de reintentos.

SkipPolicy y manejo de excepciones.

Configuración de fallos y recuperación.

Transacciones en Spring Batch

Control de transacciones en Step.

Configuración de límites de commit.

Rollback y recuperación de transacciones.

## MONITOREO Y GESTIÓN

Monitoreo y Reporting

Monitoreo de Jobs y Steps.

Configuración y uso de Spring Batch Admin.

Generación de reportes y logs.

Gestión y Mantenimiento

Reinicio y reanudación de Jobs.

Estrategias de mantenimiento.

Mejores prácticas y patrones de diseño.