CJAVA

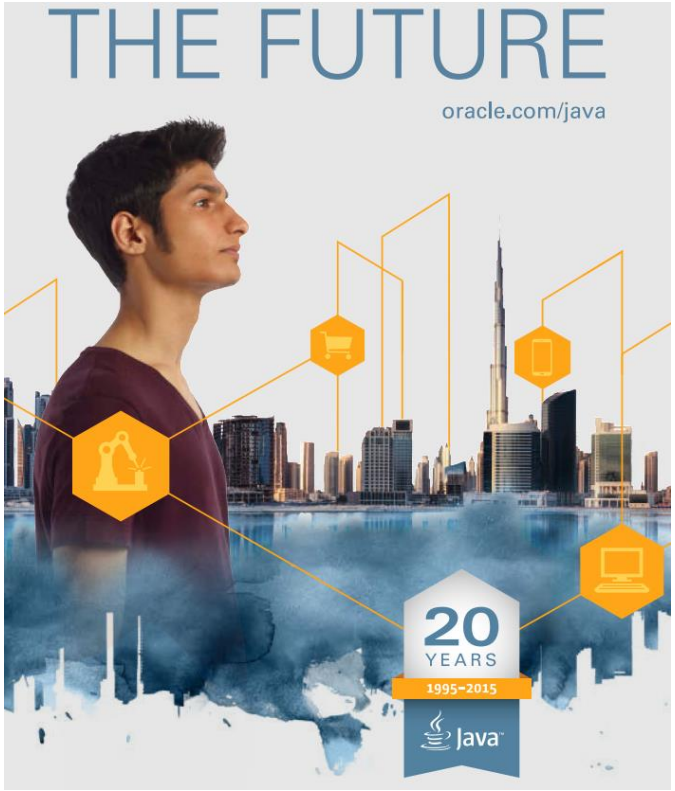
siempre para apoyarte

Visión

Poder aportar al desarrollo del País usando tecnología Java.

CJAVA
siempre para apoyarte

Quienes Somos

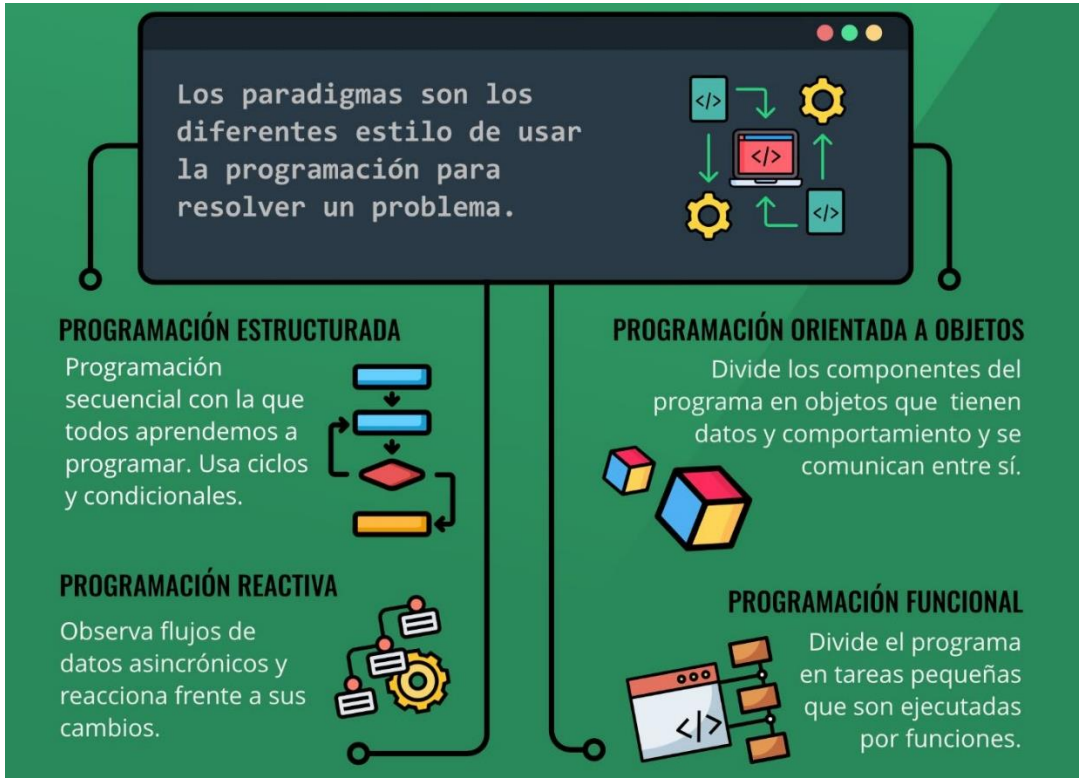


Somos una organización peruana orientada a desarrollar, capacitar e investigar tecnología JAVA a través de un prestigioso staff de profesionales a nivel nacional.

PROGRAMACION REACTIVA

Edwin maravi

Que son los paradigmas de la programación



CJAVA
siempre para apoyarte

Para que sirve?

Se enfoca en escribir código que reacciona a eventos, lo cual es útil para aplicaciones que requieren manejar múltiples eventos asincrónicos de manera eficiente.

CJAVA
siempre para apoyarte

Características:

- Asincrónica
- No bloqueante
- Orientada a eventos

CJAVA
siempre para apoyarte

Conceptos Clave

- **Reactive Streams:** Especificación para procesar flujos de datos de manera asincrónica.
- **Publisher:** Emite una secuencia de datos.
- **Subscriber:** Consume los datos emitidos por el Publisher.
- **Subscription:** Vincula al Publisher con el Subscriber.
- **Processor:** Actúa como intermediario entre Publisher y Subscriber.

CJAVA
siempre para apoyarte

Herramientas en el Ecosistema Java

- **Reactor:** Biblioteca para programación reactiva en Java.
- **RxJava:** Otra biblioteca popular para programación reactiva.

CJAVA
siempre para apoyarte

Ejemplo:

```
import reactor.core.publisher.Flux;

public class ReactiveExample {
    public static void main(String[] args) {
        Flux<String> flux = Flux.just(“Hola", “CJava", “siempre", “apoya");

        flux.subscribe(System.out::println);
    }
}
```

CJAVA
siempre para apoyarte

Operadores en Reactor

- **map:** Transforma los elementos emitidos.
- **filter:** Filtra los elementos basados en una condición.
- **flatMap:** Transforma los elementos en Publishers y los fusiona.

CJAVA
siempre para apoyarte

Ejemplo Avanzado con Operadores

```
import reactor.core.publisher.Flux;

public class EjemploReactivo {
    public static void main(String[] args) {
        Flux<String> flux = Flux.just("Hello", "World", "from", "Reactor")
                                .filter(s -> s.length() > 4)
                                .map(String::toUpperCase);

        flux.subscribe(System.out::println);
    }
}
```

CJAVA
siempre para apoyarte

Manejo de errores

onErrorResume: Proporciona un Publisher alternativo en caso de error.

onErrorReturn: Emite un valor predeterminado en caso de error.

CJAVA
siempre para apoyarte

Ejemplo de Manejo de Errores

```
import reactor.core.publisher.Flux;

public class EjemploReactivo {
   public static void main(String[] args) {
      Flux<String> flux = Flux.just("Hello", "World", "from", "Reactor")
                        .map(s -> {
                           if (s.equals("World")) {
                              throw new RuntimeException("Error occurred!");
                           }
                           return s;
                        })
                        .onErrorReturn("Fallback");

      flux.subscribe(System.out::println);
   }
}
```

CJAVA
siempre para apoyarte

Ejemplo completo

```
import reactor.core.publisher.Flux;

public class EjemploReactivo {
    public static void main(String[] args) {
        Flux<String> flux = Flux.just("apple", "banana", "cherry", "date", "elderberry")
                .filter(fruit -> fruit.length() > 5)
                .map(String::toUpperCase)
                .flatMap(fruit -> Flux.just(fruit.split("")))
                .onErrorResume(e -> Flux.just("Error", "Handling", "Fallback"))
                .doOnComplete(() -> System.out.println("Flux processing completed"));

        flux.subscribe(System.out::println,
                error -> System.err.println("Error: " + error),
                () -> System.out.println("Processing finished!"));
    }
}
```

CJAVA

siempre para apoyarte

Gracias