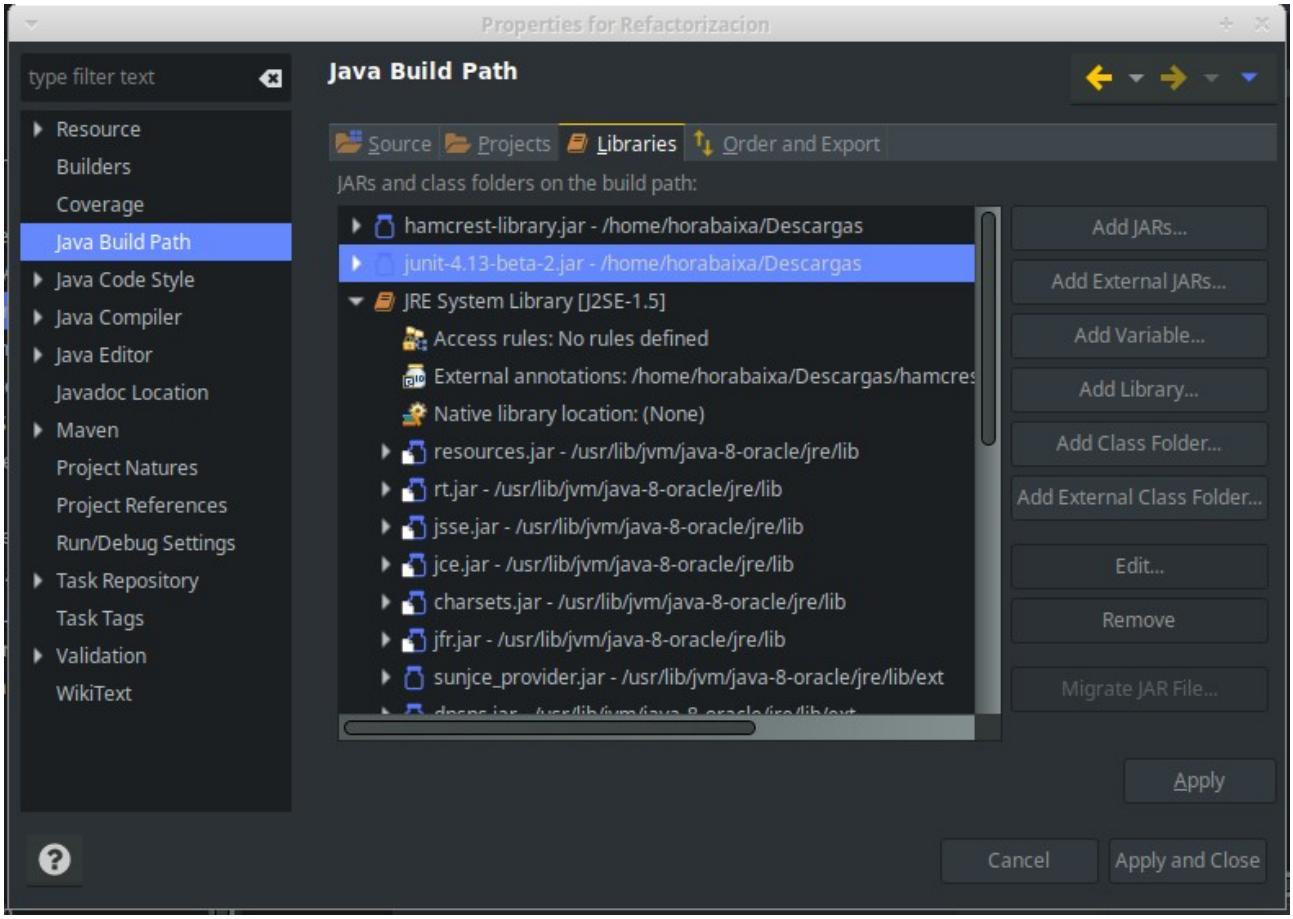
EJERCICIO PRÁCTICO – ENTORNOS DE DESARROLLO

Javier Martínez Domínguez 1º DAW

Índice

Actividades

1. Crear un nuevo proyecto en ECLIPSE (no NETBEANS), añadiendo las librerías JUNIT




```
<dependencies>
  <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
  </dependency>
  <dependency>
      <groupId>org.hamcrest</groupId>
      <artifactId>hamcrest-junit</artifactId>
      <version>2.0.0.0</version>
      <scope>test</scope>
  </dependency>
</dependencies>
/project>
```

2. Implementar y explicar detalladamente el modelo de objetos, clases y operaciones que den solución al problema anterior, así como la estructura necesaria para el testing

Clase Banco:

String compañía: nombre de la compañía
int identificador: número identificador del banco

```java
public class banco {

    String compañia;
    int identificador;

    public banco(String compañia, int identificador) {

        super();
        this.identificador = identificador;
        this.compañia = compañia;


    }
```

Clase Cliente:

String nombre: nombre del cliente
String apellidos: apellido/s del cliente
String DNI: DNI del cliente
String direccion: dirección del cliente
int telefono: teléfono de contacto del cliente

```java
public class cliente {

    String nombre;
    String apellidos;
    String DNI;
    String direccion;
    int telefono;

    public cliente(String nombre, String apellidos, String dni, String direccion, int telefono) {

        super();
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.direccion = direccion;
        this.telefono = telefono;
        DNI = dni;

    }
```

Clase cuenta:

cliente titular: guarda el cliente el cual sea titular de la cuenta
String identificador: número identificador de la cuenta
banco banco: guarda el banco de la cuenta
double saldo: salgo de la cuenta bancaria

```java
public class cuenta {

    cliente titular;
    String identificador;
    banco banco;
    double saldo;

    public cliente getTitular() {

        return titular;

    }
```

3. Crear un test que compruebe si el cliente de una cuenta (cuenta origen), es el mismo que el de otra cuenta (cuenta destino).

```java
// Comprobación Cliente

@Test
public void testCliente() {

    if (cuentaA.getTitular() == cuentaB.getTitular()) {

        System.out.println("Ok");

    } else {

        System.out.println("Error");

    }

}
```

4. Identificar mediante un test si, después de que haya realizado una transferencia de una cuenta a otra cuenta, el valor del saldo de la cuenta origen es correcto.

```
// Comprobación Salgo Original

@Test
public void testSalgoOriginal() {

    double sActual = cuentaA.getSaldo();

    cuenta.transferencia(cuentaA, cuentaB, 150);

    if (cuentaA.getSaldo() != sActual) {

        fail();

    }

}
```

5. Identificar mediante un test si, después de que haya realizado una transferencia de una cuenta a otra cuenta, el valor del saldo de la cuenta destino es correcto.

```
// Comprobación Saldo del destinatario

@Test
public void testSaldoDestinatario() {

    double sActual = cuentaB.getSaldo();

    cuenta.transferencia(cuentaA, cuentaB, 75);

    if (cuentaB.getSaldo() == sActual) {

        fail();

    }

}
```

6. Necesitamos identificar, mediante dos tests, si un cliente, desde su cuenta, que quiere realizar una transferencia de 50€ sobre otra cuenta, tiene dinero suficiente para realizarla.

```
// Comprobación Transferencia OKAY

@Test
public void testTransferenciaOk() {

    assertEquals(true, cuenta.transferencia(cuentaA, cuentaB, 50));

}

// Comprobación Transferencia ERROR

@Test
public void testTransferenciaError() {

    assertEquals(false, cuenta.transferencia(cuentaA, cuentaB, 50));

}
```

7. Identificar con un test si un cliente, desde su cuenta "00491111222233334444" quiere realizar un TRASPASO a la cuenta "00492222333344445555" de 80€, la operación se ha realizado de forma satisfactoria (verdadero).

```
// Comprobación Traspaso

@Test
public void testTraspaso() {

    if (cuentaA.getBanco() == cuentaB.getBanco() && cuentaA.getTitular() == cuentaB.getTitular()) {

        cuenta.traspaso(cuentaA, cuentaB, 80);

    } else {

        fail();

    }

}
```