LABORATORIO REFACTOR

Refactor 1 (supera la regla de las 15 lineas de longitud por método) Autor: Juan

```
public Vector<Object> getUsersMasGanancias(){
    Vector<Object>row= new Vector<Object>();
    Vector<User>vecUsers= getAllUsers();
    for(User i: vecUsers) {
        String username= i.getUserName();
        Vector<Transaccion> tran= i.getTarjeta().getTransaccioness();
        float total=0;
        int numApuestas=0;
        for(Transaccion tra: tran) {
            if(tra.isApuesta() && tra.getCant()>0) {
                total+= tra.getCant();
            } else if(tra.isApuesta() && tra.getCant()<0) {
                numApuestas++;
            }
        }
        if(total>0) {
            row.add(username);
            row.add(total);
            row.add(numApuestas);
        }
    }
    return row;
}
```

-Solución: extraer el bucle de comprobar usuarios y apuestas en una función cada bucle.

```
public Vector<Object> getUsersMasGanancias(){
    Vector<Object>row= new Vector<Object>();
    Vector<User>vecUsers= getAllUsers();
    extracted(row, vecUsers);
    return row;
}

private void extracted(Vector<Object> row, Vector<User> vecUsers) {
    for(User i: vecUsers) {
        String username= i.getUserName();
        Vector<Transaccion> tran= i.getTarjeta().getTransaccioness();
        float total=0;
        int numApuestas=0;
        for(Transaccion tra: tran) {
            if(tra.isApuesta() && tra.getCant()>0) {
                total+= tra.getCant();
            } else if(tra.isApuesta() && tra.getCant()<0) {
                numApuestas++;
            }
        }
        if(total>0) {
            row.add(username);
            row.add(total);
            row.add(numApuestas);
        }
    }
}
```

Aplicado refactor en la función extracted.

Refactor 2 (duplicidad de código) Autor: Juan
Problema: se duplica el uso de “0 goles”.
Solución: extraer el String 0 goles en una variable local.

```
vec= new Vector<String>();
vec.add(0, "0 goles");
```

4 sitios se usa este valor.

Cambio:

```
String Nogoles = "0 goles";
vec.add(0, Nogoles);
```

Refactor 3 (duplicidad de código) Autor: Juan
Problema: se duplica el uso de “1-2 goles”.
Solución: extraer el String 1-2 goles en una variable local.

```
vec.add(1, "1-2 goles");
```

Cambio:

```
String unoODosGoles = "1-2 goles";
vec.add(1, unoODosGoles);
```

Naiara Ortega, Joseba Morate, Juan Navarlaz

Refactor 4 (duplicidad de código) Autor: Juan
Problema: se duplica el uso de “empate”.
Solución: extraer el String empate en una variable local.

```
vec.add(1, "Empate");
```

Cambio:

```
String empate = "Empate";
vec.add(1, empate);
```

Refactor 5 (duplicidad de código) Autor: Joseba
Problema: se duplica el uso de “Who will score first?”.
Solución: extraer el String Who will score first? en una variable local.

```
Duplication
q2=ev1.addQuestion( 1 "Who will score first?",2);
```

Cambio:

```
String scoreFirst = "Who will score first?";
Duplication
q2=ev1.addQuestion(scoreFirst,2);
```

Refactor 6 (duplicidad de código) Autor: Joseba
Problema: se duplica el uso de "Who will win the match?".
Solución: extraer el String "Who will win the match?" en una variable local.

```
q1=ev1.addQuestion("Who will win the match?",1);
```

Cambio:

```
String winMatch = "Who will win the match?";
q1=ev1.addQuestion(winMatch,1);
```

Refactor 7 (duplicidad de código) Autor: Joseba
Problema: se duplica el uso de "Zeinek irabaziko du partidua?".
Solución: extraer el String "Zeinek irabaziko du partidua?" en una variable local.

```
Duplication
q1=ev1.addQuestion( 1 "Zeinek irabaziko du partidua?",1);
```

Cambio:

```
String zeinek = "Zeinek irabaziko du partidua?";
Duplication
q1=ev1.addQuestion(zeinek,1);
```

Refactor 8 (duplicidad de código) Autor: Joseba
Problema: se duplica el uso de "Zeinek sartuko du lehenengo gola?".
Solución: extraer el String "Zeinek sartuko du lehenengo gola?" en una variable local.

```
q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
```

Cambio:

```
String zeinek2 = "Zeinek sartuko du lehenengo gola?";
q2=ev1.addQuestion(zeinek2,2);
```

Refactor 9 (duplicidad de código) Autor: Naiara
Problema: se duplica el uso de "Fran2".
Solución: extraer el String "Fran2" en una variable local.

```
Duplication
User usuario6= new User( 1 "Fran2",
```

Cambio:

```
String fran2 = "Fran2";
Duplication
User usuario6= new User(fran2,
```

Refactor 10 (duplicidad de código) Autor: Naiara
Problema: se duplica el uso de "Fran3".
Solución: extraer el String "Fran3" en una variable local.

```
Duplication
User usuario7= new User( 1 "Fran3",
```

Cambio:

```
String fran3 = "Fran3";
Duplication
User usuario7= new User(fran3,
```

Refactor 11 (duplicidad de código) Autor: Naiara
Problema: se duplica el uso de "Juan2".
Solución: extraer el String "Juan2" en una variable local.

```
Duplication
User usuario4= new User( 1 "Juan2",
```

Cambio:

```
String juan2 = "Juan2";
Duplication
User usuario4= new User(juan2,
```

Refactor 12 (duplicidad de código) Autor: Naiara
Problema: se duplica el uso de "jaheki1761@angeleslid.com".
Solución: extraer el String "jaheki1761@angeleslid.com" en una variable local.

```
,  1 "jaheki1761@angeleslid.com");
```

Cambio:

```
String correo1 = "jaheki1761@angeleslid.com";
Duplication
User usuario2= new User("Juan", "123", "1111111111111112", correo1);
```