

# Laboratorio refactorización

## Extract Local Variable

Consiste en extraer una expresión y asignársela a una variable de ámbito local, de esta forma se sustituyen todas las apariciones de la expresiones en el ámbito local por la nueva variable.

```

GameBoardFull.java
36 PropertyCell g1 = new PropertyCell();
37 PropertyCell g2 = new PropertyCell();
38 IOwnable cc3 = new CardCell(Card.TYPE_CC, "Comm
39 PropertyCell g3 = new PropertyCell();
40 RailRoadCell rr4 = new RailRoadCell();
41 IOwnable c3 = new CardCell(Card.TYPE_CHANCE, "C
42 PropertyCell db1 = new PropertyCell();
43 PropertyCell db2 = new PropertyCell();
44 PropertyCell db3 = new PropertyCell();
45
46
47 dp1.setPrice(60);
48 dp1.setColorGroup("purple");
49 dp1.setHousePrice(50);
50 dp1.setName("Mediterranean Avenue");
51 dp1.setRent(2);
52
53 dp2.setPrice(60);
54 dp2.setColorGroup("purple");
55 dp2.setHousePrice(50);
56 dp2.setName("Baltic Avenue");
57 dp2.setRent(4);

```

```

GameBoardFull.java
36 PropertyCell g1 = new PropertyCell();
37 PropertyCell g2 = new PropertyCell();
38 IOwnable cc3 = new CardCell(Card.TYPE_CC, "Comm
39 PropertyCell g3 = new PropertyCell();
40 RailRoadCell rr4 = new RailRoadCell();
41 IOwnable c3 = new CardCell(Card.TYPE_Ch
42 PropertyCell db1 = new PropertyCell();
43 PropertyCell db2 = new PropertyCell();
44 PropertyCell db3 = new PropertyCell();
45
46
47 dp1.setPrice(60);
48 String colorPurple = "purple";
49 dp1.setColorGroup(colorPurple);
50 dp1.setHousePrice(50);
51 dp1.setName("Mediterranean Avenue");
52 dp1.setRent(2);
53
54 dp2.setPrice(60);
55 dp2.setColorGroup(colorPurple);
56 dp2.setHousePrice(50);
57 dp2.setName("Baltic Avenue");
58 dp2.setRent(4);

```

En la clase GameBoardFull se sustituyen la aparición de la cadena “purple” por una variable local de forma que si el valor de esta cadena se necesitara modificar en el futuro bastaría con hacerlo una sola vez, al crear la variable.

## Extract constant

Al igual que con la opción de Extract Local Variable se sustituye la aparición de una expresión por una variable. En este caso la nueva variable tiene un ámbito global y es constante, por lo tanto el valor no puede modificarlo ninguna función.

```

FreeParkingCell.java
1 package edu.ncsu.monopoly;
2
3 public class FreeParkingCell extends Cell {
4
5     public FreeParkingCell() {
6         setName("Free Parking");
7     }
8
9     public void playAction() {
10         return;
11     }
12 }

```

```

FreeParkingCell.java
1 package edu.ncsu.monopoly;
2
3 public class FreeParkingCell extends Cell {
4
5     private static final String FREE_PARKING = "Free Parking";
6
7     public FreeParkingCell() {
8         setName(FREE_PARKING);
9     }
10
11     public void playAction() {
12         return;
13     }
14 }
15

```


En FreeParkingCell se ha sustituido el String “Free Parking” por una variable constante de ámbito global a la que se podría acceder desde cualquier parte de la clase aunque en este caso no sea necesario.

## Inline

Consiste en juntar en una sola línea de código lo que esta hecho en en dos, se sustituye la referencia a una variable por el valor de esta.

```
*JailCard.java  ⌕
1 package edu.ncsu.monopoly;
2
3
4 public class JailCard extends Card {
5     int type;
6
7     public JailCard(int cardType) {
8         type = cardType;
9     }
10
11     public void applyAction() {
12         Player currentPlayer = GameMaster.instance().getCurrentPlayer();
13         JailCell jail = (JailCell)(GameMaster.instance().getGameBoard().queryCell("Jail"));
14         GameMaster.instance().sendToJail(currentPlayer);
15     }
16 }

*JailCard.java  ⌕
1 package edu.ncsu.monopoly;
2
3
4 public class JailCard extends Card {
5     int type;
6
7     public JailCard(int cardType) {
8         type = cardType;
9     }
10
11     public void applyAction() {
12         JailCell jail = (JailCell)(GameMaster.instance().getGameBoard().queryCell("Jail"));
13         GameMaster.instance().sendToJail(GameMaster.instance().getCurrentPlayer());
14     }
15 }
```



En JailCard se ha sustituido la variable currentPlayer por su valor cuando se llama a la función sendToJail() y queda todo en una única línea de código.

## Convert local variable to field

Se trata de modificar el alcance de una variable, de local pasa a ser un atributo privado de la clase.

```

Player.java
1 package edu.ncsu.monopoly;
2
3 import java.util.ArrayList;
4
5
6
7
8 public class Player {
9     //the key of colorGroups is the name of the color group.
10    private Hashtable colorGroups = new Hashtable();
11    private boolean inJail;
12    private int money;
13    private String name;
14
15    private IOwnable position;
16    private ArrayList properties = new ArrayList();
17    private ArrayList railroads = new ArrayList();
18    private ArrayList utilities = new ArrayList();
19
20    public Player() {
21        GameBoard gb = GameMaster.instance().getGameBoard();
22        inJail = false;
23        if(gb != null) {
24            position = gb.queryCell("Go");
25        }
26    }
27 }

Player.java
2
3 import java.util.ArrayList;
4
5
6
7
8 public class Player {
9     //the key of colorGroups is the name of the color group.
10    private Hashtable colorGroups = new Hashtable();
11    private boolean inJail;
12    private int money;
13    private String name;
14
15    private IOwnable position;
16    private ArrayList properties = new ArrayList();
17    private ArrayList railroads = new ArrayList();
18    private ArrayList utilities = new ArrayList();
19    private GameBoard gb = GameMaster.instance().getGameBoard();
20
21    public Player() {
22        inJail = false;
23        if(gb != null) {
24            position = gb.queryCell("Go");
25        }
26    }
27 }
  
```

En el ejemplo, el constructor crea una variable del tipo GameBoard que se utiliza en diferentes partes de la clase repitiendo el proceso de crearla. Después de refactorizar, esa variable se convierte en un atributo al que pueden acceder todas las funciones.

## Extract superclass

Se crea una superase con los métodos que, a elección del programador, contiene la clase original. Si existiera ya una superclase se modificaría para que la nueva pasase a ser la superclase de la clase original y la superclase antigua sería superclase de la nueva superclase. En caso de que la superclase utilizara atributos propios de la clase original que no se incluyesen daría errores de compilación.

```

Player.java
54 }
55
56 public boolean checkProperty(String property) {
57     for(int i=0;i<properties.size();i++) {
58         IOwnable cell = (IOwnable)properties.get(i);
59         if(cell.getName().equals(property)) {
60             return true;
61         }
62     }
63     return false;
64 }
65 }
66

SuperPlayer.java
1 package edu.ncsu.monopoly;
2
3 import java.util.ArrayList;
4
5 public class SuperPlayer {
6
7     protected ArrayList properties = new ArrayList();
8
9     public SuperPlayer() {
10         super();
11     }
12
13     public boolean checkProperty(String property) {
14         for(int i=0;i<properties.size();i++) {
15             IOwnable cell = (IOwnable)properties.get(i);
16             if(cell.getName().equals(property)) {
17                 return true;
18             }
19         }
20         return false;
21     }
22 }
23
24 }
  
```

De la clase Player se ha extraído una superclase llamada SuperPlayer de la que Player hereda el método checkProperty y el atributo property que antes de la refactorización estaban en Player.

## Infer generic type argument

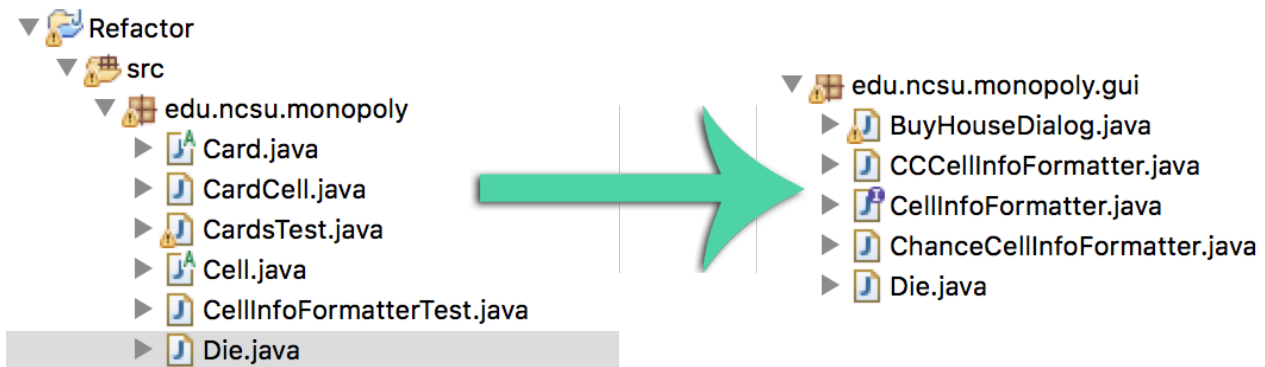
Permite especificar los tipos correctos de argumentos para las clases, de forma que cuando se utilizan clases genéricas se especifique que tipo se va a utilizar.



En el ejemplo inicialmente cuando se crea la variable `players` no se especifican de que tipo serán los objetos que se almacenarán en el `arraylist`, esto permite que se pueda almacenar cualquier cosa que podría causar algún error si se utiliza erróneamente. Después de refactorizar solo se permiten almacenar objetos de tipo `Player`.

## Move

La opción `move` permite trasladar una clase de un paquete a otro.



La clase `Die` que originalmente estaba en el paquete `Monopoly` se ha trasladado a `Monopoly.gui` utilizando la opción `move` en el menú de refactorizar.