# Извлечение метода (Extract Method)

## Описание

Применяется, если некоторая часть кода требует комментария для описания своей функциональности. Тогда имеет смысл вынести этот код в отдельный метод. Это приведет к тому, что код, из которого производится извлечение, становится похожим на набор содержательных комментариев без излишних подробностей. Также вызов метода, содержащего извлеченный код можно использовать в других местах.

## Участок кода до рефакторинга

void Controller::makeStep(int hole)

{

...

***mainDlg->setTip("Ход игрока " + playersList- >getActivePlayer()>getName());***

***mainDlg->ActivateButtons(TRUE);***

***for (int i = 0; i < 6; i++)***

***if ( netStruc->isPositionBlocked(i) )***

***{***

***CButton button = mainDlg->GetDlgItem(IDC\_BUTTON\_HOLE1 + i);***

***ATLASSERT(::IsWindow(button.m\_hWnd));***

***button.EnableWindow(FALSE);***

***}***

...

}

## Участок кода после рефакторинга

void Controller::makeStep(int hole)

{

...

***//! Делаем неактивными кнопки, если нельзя кинуть фишку в их отверстие***

***deActivateButtons();***

...

}

***//! Деактивирует кнопки для дырок, в которые нельзя кинуть фишки***

***void Controller::deActivateButtons()***

***{***

***//! Сменили подсказку***

***mainDlg->setTip("Ход игрока " +***

***playersList->getActivePlayer()->getName());***

***mainDlg->ActivateButtons(TRUE);***

***for (int i = 0; i < 6; i++)***

***if ( netStruc->isPositionBlocked(i) )***

***{***

***CButton button = mainDlg->GetDlgItem(IDC\_BUTTON\_HOLE1 + i);***

***ATLASSERT(::IsWindow(button.m\_hWnd));***

***button.EnableWindow(FALSE);***

***}***

***}***

# Извлечение метода (Extract Method)

## Описание

Применяется, если некоторая часть кода требует комментария для описания своей функциональности. Тогда имеет смысл вынести этот код в отдельный метод. Это приведет к тому, что код, из которого производится извлечение, становится похожим на набор содержательных комментариев без излишних подробностей. Также вызов метода, содержащего извлеченный код можно использовать в других местах.

## Участок кода до рефакторинга

// ==========================================================================

// ЗАПУСТИТЬ ВСЕ ДОПУСТИМЫЕ СРАБАТЫВАНИЯ ПЕРЕХОДОВ

template <typename MT>

inline bool PetriNet<MT>::FireAllTransitions(iface::iDrawable\* drawable = NULL)

{

itransition itF = GetTransitions(); // Исходный итератор

TRANSITION fired = 0; // Кол-во сработавших переходов

itransition it = itF;

while ( !it.end() && (fired < m\_maxFired) )

// ПЕРЕХОД АКТИВЕН

if ( IsTransitionActive(it.transition()) )

{

***if ( IsTransitionActive(tr) )***

***{***

***// ЗАБАРАТЬ ФИШКИ ИЗ ВХОДНЫХ ПОЗИЦИЙ***

***itransition\_input input = GetTransitionInput(tr);***

***for ( ; !input.end(); ++input )***

***SetToken(input.position(), false);***

***// ПЕРКИНУТЬ В ВЫХОДНЫЕ***

***itransition\_output output = GetTransitionOutput(tr);***

***for ( ; !output.end(); ++output )***

***SetToken(output.position(), true);***

***}***

// УВЕЛИЧИТЬ СЧЕТЧИК

fired++;

// ВЕРНУТСЯ В НАЧАЛО ОБХОДА

it = itF;

// ВЫЗВАТЬ ПЕРЕРИСОВКУ

if ( drawable != NULL )

drawable->Redraw();

}

// ПЕРЕХОД НЕАКТИВЕН

else

++it;

if ( fired == m\_maxFired )

return false;

else

return true;

}

## Участок кода после рефакторинга

// ==========================================================================

// ЗАПУСТИТЬ ВСЕ ДОПУСТИМЫЕ СРАБАТЫВАНИЯ ПЕРЕХОДОВ

template <typename MT>

inline bool PetriNet<MT>::FireAllTransitions(iface::iDrawable\* drawable = NULL)

{

itransition itF = GetTransitions(); // Исходный итератор

TRANSITION fired = 0; // Кол-во сработавших переходов

itransition it = itF;

while ( !it.end() && (fired < m\_maxFired) )

// ПЕРЕХОД АКТИВЕН

if ( IsTransitionActive(it.transition()) )

{

***// ЗАПУСТИТЬ СРАБАТЫВАНИЕ***

***FireTransition(it.transition());***

// УВЕЛИЧИТЬ СЧЕТЧИК

fired++;

// ВЕРНУТСЯ В НАЧАЛО ОБХОДА

it = itF;

// ВЫЗВАТЬ ПЕРЕРИСОВКУ

if ( drawable != NULL )

drawable->Redraw();

}

// ПЕРЕХОД НЕАКТИВЕН

else

++it;

if ( fired == m\_maxFired )

return false;

else

return true;

}

// ==========================================================================

// ЗАПУСТИТЬ СРАБАТЫВАНИЕ ЗАДАННОГО ПЕРЕХОДА

/\* Так как поведение зависит от типа маркировки, функция

специфицируется отдельно для булевого и числовых типов \*/

inline void PetriNet<bool>::FireTransition(TRANSITION tr)

{

if ( !IsTransitionActive(tr) )

return;

// ЗАБАРАТЬ ФИШКИ ИЗ ВХОДНЫХ ПОЗИЦИЙ

itransition\_input input = GetTransitionInput(tr);

for ( ; !input.end(); ++input )

SetToken(input.position(), false);

// ПЕРКИНУТЬ В ВЫХОДНЫЕ

itransition\_output output = GetTransitionOutput(tr);

for ( ; !output.end(); ++output )

SetToken(output.position(), true);

}

# Переименование метода (Rename Method)

## Описание

Применяется, если существует метод, по названию которого не догадаться о его назначении. Необходимо переименовать метод.

## Применение

Для интерфейса iNetworkStruct метод с названием clean(), означающий, судя по названию, скорее проверку, чем действие, переименовывается на clear(), что больше воспринимается как «очистить».

***//! Очистить структуру от фишек***

***virtual void clean () = 0;***

***//! Очистить структуру от фишек***

***virtual void clear () = 0;***