

Пример разработки пользовательского интерфейса.

В качестве примера рассмотрим web-браузер. web-браузер - это пользовательское приложение с активным использованием интерфейса пользователя, поэтому он должен быть детально продуман с целью максимального упрощения и удобства работы с приложением. В качестве модели для дальнейшего рассмотрения возьмём основное окно web-браузера.

Назначение :

Основное окно web-браузера используется пользователем для навигации страницами интернет ресурсов.

Начальное описание потока событий работы с web-браузером.

1. Прецедент начинается когда пользователь запускает приложение web-браузера.
2. Пользователь может выполнить одно из следующих действий:
 1. Ввести ссылку интересующего его ресурса и перейти по ней.
 2. Перейти на домашнюю страничку.
 3. Перейти на предыдущую страницу.
 4. Перейти на следующую страницу.
 5. Остановить загрузку страницы.
 6. Обновить ресурс.
3. Прецедент заканчивается, когда пользователь завершает работу с приложением.

Разработка ориентиров:

1. Прецедент начинается когда пользователь запускает приложение web-браузера. [При запуске приложения фокус уместно поместить в поле ввода имени ресурса. Пользователь должен иметь возможность видеть статус текущей страницы(загрузка/готово/открытие/....).]
2. Пользователь может выполнить одно из следующих действий
 1. Ввести ссылку интересующего его ресурса и перейти по ней.[Пользователь должен иметь возможность видеть список ранее запрашиваемых им узлов.]
 2. Перейти на домашнюю страничку.
 3. Перейти на предыдущую страницу.[Пользователь должен иметь возможность видеть список ранее посещённых им узлов.]
 4. Перейти на следующую страницу.[Пользователь должен иметь возможность видеть список ранее посещённых им узлов.]
 5. Остановить загрузку страницы.[Пользователь должен иметь возможность остановить загрузку текущей страницы.]
 6. Обновить ресурс.[Пользователь должен иметь возможность обновить текущую страницу.]

Разработка атрибутов:

1. Прецедент начинается когда пользователь запускает приложение web-браузера.
2. Пользователь может выполнить одно из следующих действий
 1. Ввести ссылку интересующего его ресурса и перейти по ней.{Текст ссылки в среднем содержит до 100 символов}
 2. Перейти на домашнюю страничку.
 3. Перейти на предыдущую страницу.{История предыдущих страниц в среднем не содержит более 10 страниц}
 4. Перейти на следующую страницу.{История предыдущих страниц в среднем не содержит более 10 страниц}
 5. Остановить загрузку страницы.{Остановка загрузки должна происходить, по мнению клиента, моментально. В 20 % случаев пользователю необходимо остановить загрузку текущей страницы.}
 6. Обновить ресурс.{В 25% случаев пользователю необходимо перегрузить текущую страницу.}

Разработка интенсивности использования:

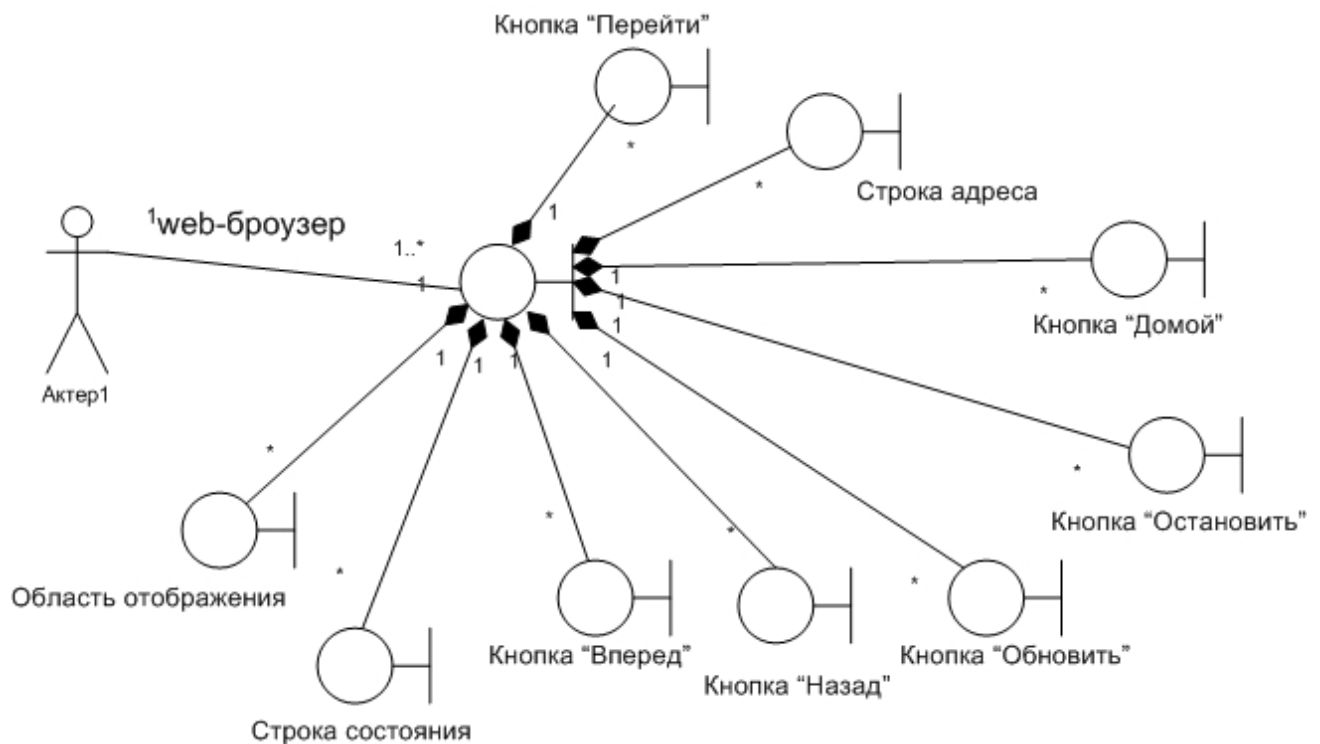
1. Прецедент начинается когда пользователь запускает приложение web-браузера.
2. Пользователь может выполнить одно из следующих действий
 1. Ввести ссылку интересующего его ресурса и перейти по ней.(Выполняется больше чем в 65 % случаев.)
 2. Перейти на домашнюю страничку. (Выполняется менее чем в 5 % случаев.)
 3. Перейти на предыдущую страницу.(Выполняется больше чем в 35 % случаев.)
 4. Перейти на следующую страницу.(Выполняется менее чем в 10 % случаев.)
 5. Остановить загрузку страницы.(Выполняется больше чем в 35 % случаев.)
 6. Обновить ресурс.(Выполняется менее чем в 25 % случаев.)

Результирующие описание потока событий примера :

1. Прецедент начинается когда пользователь запускает приложение web-браузера.[При запуске приложения фокус уместно поместить в поле ввода имени ресурса. Пользователь должен иметь возможность видеть статус текущей страницы(загрузка/готово/открытие/....).]
2. Пользователь может выполнить одно из следующих действий:
 1. Ввести ссылку интересующего его ресурса и перейти по ней.[Пользователь должен иметь возможность видеть список ранее запрашиваемых им узлов.]{Текст ссылки в среднем содержит до 100 символов}{Выполняется больше чем в 65 % случаев.)}

2. Перейти на домашнюю страничку.(Выполняется менее чем в 5 % случаев.)
 3. Перейти на предыдущую страницу.[Пользователь должен иметь возможность видеть список ранее посещённых им узлов.]{История предыдущих страниц в среднем не содержит более 10 страниц}(Выполняется больше чем в 35 % случаев.)
 4. Перейти на следующую страницу.[Пользователь должен иметь возможность видеть список ранее посещённых им узлов.]{История предыдущих страниц в среднем не содержит более 10 страниц}(Выполняется менее чем в 10 % случаев.)
 5. Остановить загрузку страницы.[Пользователь должен иметь возможность остановить загрузку текущей страницы.]{Остановка загрузка должна происходить, по мнению клиента, моментально. В 20 % случаев пользователю необходимо остановить загрузку текущей страницы.}(Выполняется больше чем в 35 % случаев.)
 6. Обновить ресурс.[Пользователь должен иметь возможность обновить текущую страницу.]{В 25% случаев пользователю необходимо перегрузить текущую страницу.}(Выполняется менее чем в 25 % случаев.)
3. Прецедент заканчивается, когда пользователь завершает работу с приложением.

Диаграммы граничных классов.

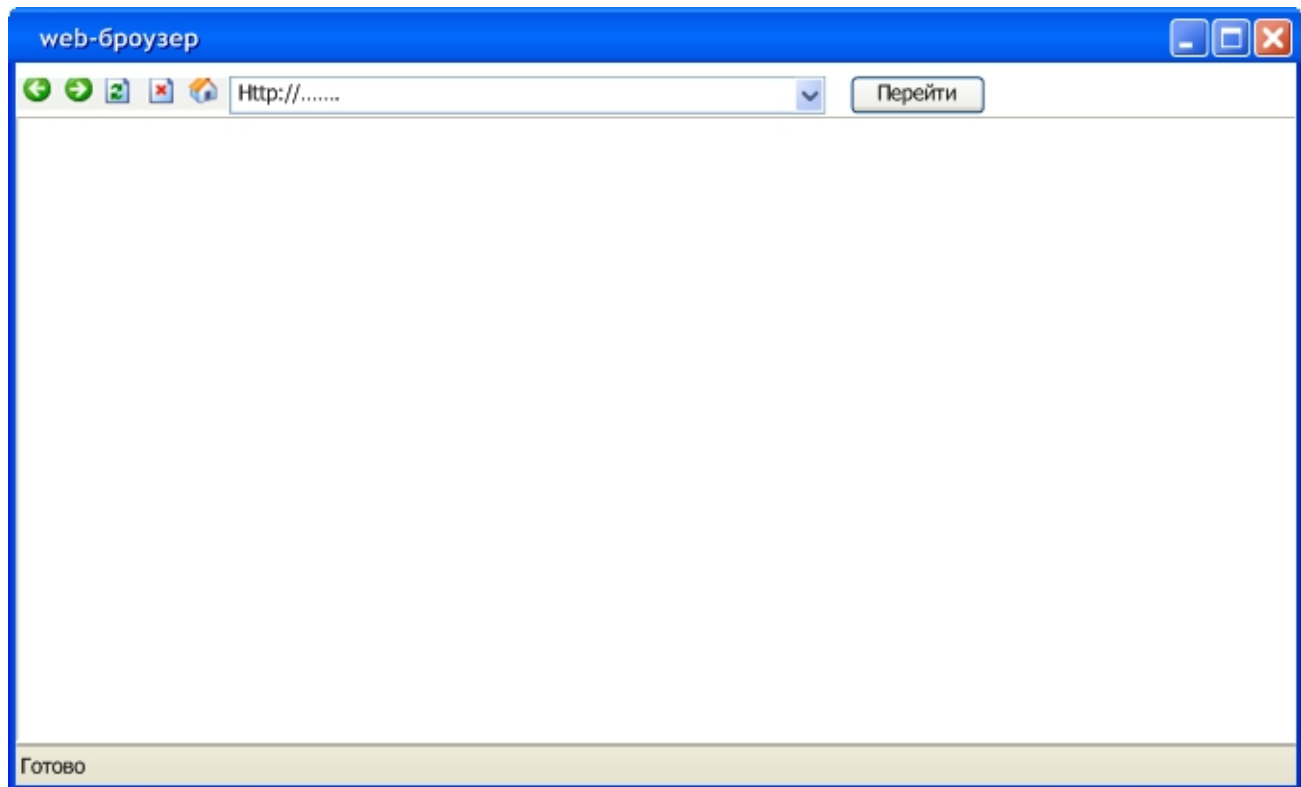


Дополнение описания иллюстрированного сценария объектами диаграмм.

1. Прецедент начинается когда пользователь запускает приложение web-браузера. [При запуске приложения фокус уместно поместить в поле ввода имени ресурса. Пользователь должен иметь возможность видеть статус текущей страницы(загрузка/готово/открытие/....).] "web-браузер"
2. Пользователь может выполнить одно из следующих действий:
 1. Ввести ссылку интересующего его ресурса и перейти по ней. [Пользователь должен иметь возможность видеть список ранее запрашиваемых им узлов.] {Текст ссылки в среднем содержит до 100 символов} {Выполняется больше чем в 65 % случаев.} "Строка адреса" "Кнопка Перейти"
 2. Перейти на домашнюю страничку. {Выполняется менее чем в 5 % случаев.} "Кнопка Домой"
 3. Перейти на предыдущую страницу. [Пользователь должен иметь возможность видеть список ранее посещённых им узлов.] {История предыдущих страниц в среднем не содержит более 10 страниц} {Выполняется больше чем в 35 % случаев.} "Кнопка Назад"
 4. Перейти на следующую страницу. [Пользователь должен иметь возможность видеть список ранее посещённых им узлов.] {История предыдущих страниц в среднем не содержит более 10 страниц} {Выполняется менее чем в 10 % случаев.} "Кнопка Вперёд"

5. Остановить загрузку страницы. [Пользователь должен иметь возможность остановить загрузку текущей страницы.] {Остановка загрузка должна происходить, по мнению клиента, моментально. В 20 % случаев пользователю необходимо остановить загрузку текущей страницы.} (Выполняется больше чем в 35 % случаев.) "Кнопка Остановить"
6. Обновить ресурс. [Пользователь должен иметь возможность обновить текущую страницу.] {В 25% случаев пользователю необходимо перегрузить текущую страницу.} (Выполняется менее чем в 25 % случаев.) "Кнопка Обновить"
3. Прецедент заканчивается, когда пользователь завершает работу с приложением.

Приблизительное геометрическое расположение компонентов приложения на экране.



При разработке графического интерфейса пользователя необходимо понимать что интерфейс, прежде всего, должен быть понятен для конечного пользователя продукта с его точки зрения. Ведь только пользователь знает, что он ожидает от приложения и как ему удобно работать с системой.

Пользовательский интерфейс должен подчеркивать основные возможности системы: наиболее часто употребляемые функции должны быть вынесены в панель

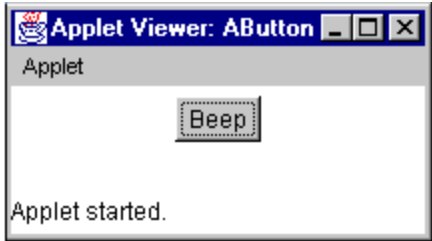
инструментов или в пункты меню. Пользователь не должен искать где он может выполнить то или иное действие. Все должно быть интуитивно понятно и быстро достигаемо. Необходимо постараться сократить количество переходов по пунктам меню или мастерам для совершения той или иной операции. Например, сложно себе представить браузер, в котором для открытия новой странички необходимо выбрать в меню мастер (wizard), который состоит из нескольких шагов, для открытия новой странички. Любая операция должна решаться за один-два перехода пользователя. В современных интерфейсах огромное значение также играют мелкие детали и подсказки пользователю. С продуктом намного удобнее работать, если ты в затруднительный для себя момент с легкостью можешь увидеть подсказку, при наведении на элементы можно увидеть описание и цель использования данных управляющих элементов. В сообщениях также необходимо стараться подчеркнуть главное. Наиболее значимое содержимое можно выделить курсивом или жирным шрифтом, при наведении на кнопку или ссылку можно изменить курсор мыши на указатель и т.п.

Пользователь также всегда намного проще воспринимает графическую информацию чем текстовое описание. При выборе между таблицей и графиком всегда отдавайте предпочтение графику так как это более наглядно и требует меньше времени для понимания. Необходимо стараться избегать многословности и перегруженности интерфейса множеством данных или элементов. Таблица из 1000 строк принесет больше вреда чем пользы!, а среди 25 кнопок пользователь обязательно запутается.

Интерфейс должен удерживать пользователя в рамках, на которые рассчитывает программа – он должен подсказывать дальнейшие действия и не давать возможности выполнить что-либо, что может принести вред системе или пользователю.

И помните, что большинство пользователей вашей системы не будут иметь представления о том, как с ней работать, они никогда не будут читать документацию, и не будут хотеть разобраться в возникших проблемах без помощи системы.

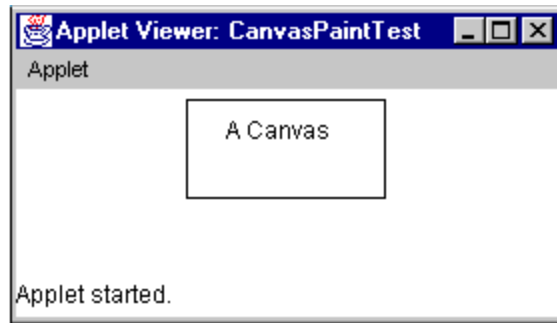
Java AWT Controls

<p>java.awt.Button:</p> <pre>import java.awt.Button; import java.applet.Applet; public class AButton extends Applet { public void init() { Button beepButton = new Button("Beep"); add(beepButton); } }</pre>	 <p>Используется для отображения кнопок.</p>
<p>java.awt.Canvas:</p> <pre>import java.awt.Canvas; import java.awt.Graphics;</pre>	

```

class DrawingRegion extends Canvas {
public DrawingRegion() {
setSize(100, 50);
}
public void paint(Graphics g) {
g.drawRect(0, 0, 99, 49);
g.drawString("A Canvas", 20,20);
}
}

```



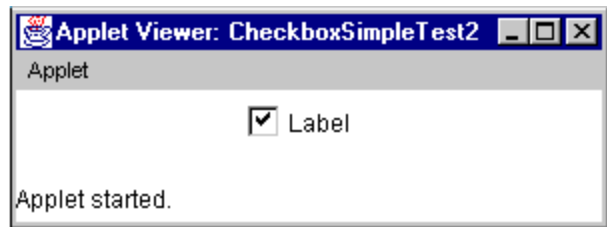
Используется для отображения канвы, на которой можно рисовать графику, в основном.

```

java.awt.Checkbox:
import java.awt.*;
import java.applet.Applet;

public class CheckboxSimpleTest2 extends Applet {
public void init() {
Checkbox m = new Checkbox("Label",
true);
add(m);
}
}

```



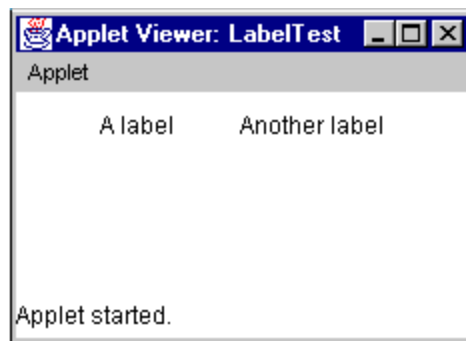
Используется для отображения выбора опций

```

java.awt.Label:
import java.awt.*;
import java.applet.Applet;

public class LabelTest extends Applet {
public void init() {
add(new Label("A label"));
add(new Label("Another label",
Label.RIGHT));
}
}

```



Используется для отображения надписей

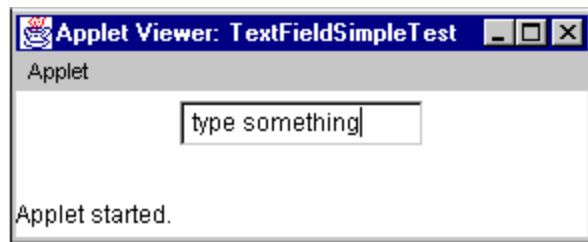
```

java.awt.TextField:
import java.awt.*;
import java.applet.Applet;

public class TextFieldSimpleTest extends

```

```
Applet {
public void init() {
TextField f1 =
new TextField("type something");
add(f1);
}
}
```

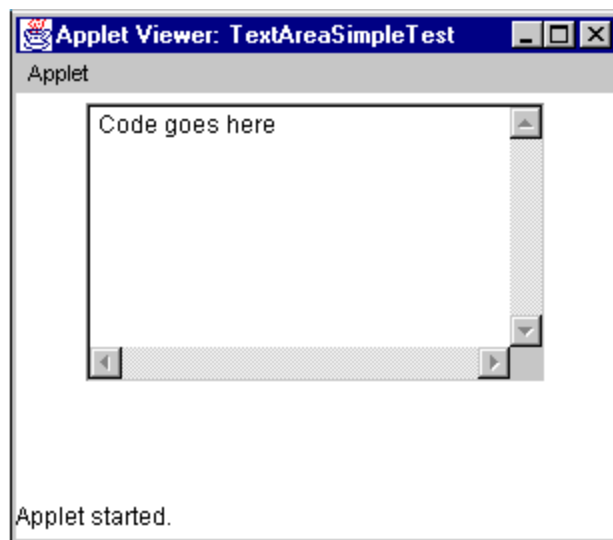


Используется для отображения текстовых полей. С одной линией для ввода текста

java.awt.TextArea:

```
import java.awt.*;
import java.applet.Applet;

public class TextAreaSimpleTest extends Applet {
TextArea disp;
public void init() {
disp = new TextArea("Code goes here", 10, 30);
add(disp);
}
}
```



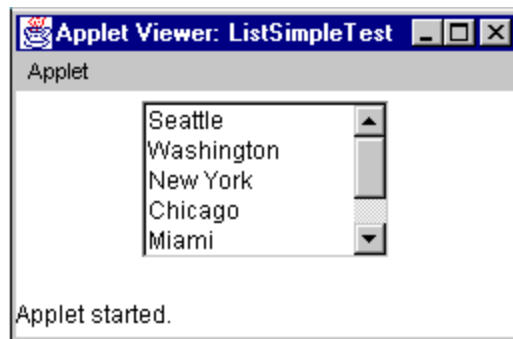
Используется для отображений областей для ввода текста.

java.awt.List:

import java.awt.*;

```
import java.applet.Applet;
```

```
public class ListSimpleTest extends Applet {
public void init() {
List list = new List(5, false);
list.add("Seattle");
list.add("Washington");
list.add("New York");
list.add("Chicago");
list.add("Miami");
list.add("San Jose");
}
```



Используется для отображения списков со значениями.

list.add("Denver"); add(list); } }	
---	--

В данном примере для реализации можно использовать:

web-броузер	java.applet.Applet/java.awt.Frame
"Строка адреса"	java.awt.TextField
"Кнопка Перейти"	java.awt.Button
"Кнопка Домой"	java.awt.Button
"Кнопка Назад"	java.awt.Button
"Кнопка Вперёд"	java.awt.Button
"Кнопка Остановить"	java.awt.Button
"Кнопка Обновить"	java.awt.Button