ОБРАБОТКА СОБЫТИЙ

Назначить обработчик событий

- Методом с названием setOnXXX, изменяющим свойство onXXX

```
Button bt = new Button (“Press”);
bt.setOnAction( event -> {
        System.out.println(“Нажата кнопка Press”);
});
```

Отменить обработчик

```
bt.setOnAction(null);
```

Назначить обработчик событий

- Методами addEventFilter() и addEventHandler()
- Можно удалять removeEventFilter() и removeEventHandler()

- Порядок срабатывания для Button

  Filter

  Handler

  setOnAction

Назначить обработчик событий

- addEventFilter() можно зарегистрировать несколько

```
bt.addEventFilter(ActionEvent.ACTION, event->{
        System.out.println(“Нажата кнопка Press”);
        if ( ???) {
                System.out.println(“Больше не нажимать”);
                event.consume();
        }
});
```

Назначить обработчик событий

- addEventHandler() можно зарегистрировать несколько

```
bt.addEventHandler(ActionEvent.ACTION, event->{
        System.out.println(“Нажата кнопка Press”);
});
```

Назначить обработчик событий

- Если нужно отследить изменения свойства

```
stage.xPropert().addListener((obj, oldV, newV) ->{
        System.out.println(“Движемся по X ”+newV);
});
```

В метод регистрации можно передать

- Анонимный вложенный класс, реализующий интерфейс EventHandler

```
bt.setOnAction( new EventHandler<ActionEvent>()
@Override
 public void handle(ActionEvent event) {
        System.out.println(“Нажата кнопка Press”);
});
```

- Лямбда-выражение

```
bt.setOnAction( event -> {
        System.out.println(“Нажата кнопка Press”);
});
```

В метод регистрации можно передать

- Объект класса, реализующего интерфейс EventHandler

```
private EventHandler<ActionEvent> handl1 = event -> {
        System.out.println(“Нажата кнопка Press”);
};
bt.setOnAction(handl1);
```

- Ссылку на метод, получающий объект- событие XXX

```
private
 void onClick(ActionEvent event){
        System.out.println(“Нажата кнопка Press”);
};
bt.setOnAction(this::onClick);
```

Движение события

- Методы addEventFilter() и addEventHandler() имеют все узлы, объекты сцены и окна
- Если назначить обработчик нажатия кнопки для контейнера, то он перехватит нажатие всех кнопок в контейнере
- Событие движется сначала от окна до источника, вызвавшего события, а затем обратно к окну

Пример

- Создаем приложение с одной кнопкой на сцене, компоновка - любая
- Кнопка, контейнер, сцена и окно имеют обработчики событий Filter, Handler. Кнопка еще имеет обработчик OnAction
- Протокол вызовов будет таким:

  stage – Filter

  scene – Filter

  root – Filter

  button – Filter

  button – Handler

  button – OnAction

  root – Handler

  scene – Handler

  stage – Handler

Пример

Создаем сцену с одной кнопкой

```
VBox root = new VBox(15.0);
 root.setAlignment(Pos.CENTER);
 Button button = new Button("Нажми меня");
 root.getChildren().addAll(button);
 Scene scene = new Scene(root, 500.0, 150.0);
```

Пример

- Добавляем обработчики

```
button.setOnAction(event -> {
        System.out.println("button - setOnAction");
});
     button.addEventHandler(ActionEvent.ACTION, event -> {
        System.out.println("button - Handler");
}); // добавить еще
button.addEventFilter(ActionEvent.ACTION, event -> {
        System.out.println("button - Filter");
}); // добавить еще
```

Пример

- Добавляем обработчики EventHandler и EventFilter
  - *Графу сцены root*
  - *Сцене scene*
  - *Окну stage*
- Проверяем последовательность срабатывания
- Добавляем в обработчики (не все) отмену обработки после первого срабатывания

```
event.consume();
```

Генерация события

- Если класс реализует интерфейс EventTarget, ему можно послать событие программным путем

```
bt.setOnAction( event -> {
        System.out.println(“Нажата кнопка Press”);
        System.out.println(“генерируем событие нажатия другой кнопки”);
        Event.fireEvent(otherBt, event);
});
```

Пример

- Добавляем на сцену еще одну кнопку.
- Назначаем у нее обработчик
- Делаем передачу события от этой кнопки, первой кнопке (или наоборот)

Классы событий

ActionEvent

InputEvent

- KeyEvent
- MouseEvent
  - MouseDragEvent
- TouchEvent
- GestureEvent
- DragEvent
- ContexMenuEvent

WindowEvent

События окна

- WindowEvent
  - WindowEvent. ANY
  - WindowEvent. WINDOW_SHOWING
  - WindowEvent. WINDOW_SHOWN
  - WindowEvent. WINDOW_CLOSE_REQUEST
  - WindowEvent. WINDOW_HIDING
  - WindowEvent. WINDOW_HIDDEN

События окна

```
stage.addEventHandler ( WindowEvent.ANY,
                        event->
                        { System.out.println(“ANY –”+ event.getEventType();}
);
```

События окна

- Свойства, отражающие изменение состояния окна
  stage.showingProperty()
  stage.focusedProperty()
  stage.xProperty()
  stage.yProperty()
  stage.widthProperty()
  stage.heightProperty()
  stage.iconofiedProperty()
  stage.maximisedProperty()
  stage.fullScreenProperty()

События окна

```
stage.xProperty.addListner ( (obj, oldV, newV )->
                        { System.out.println(“Coord X - ”+ oldV +” new –”+ newV);}
);
```

События клавиатуры

- KeyEvent
  - KeyEvent.ANY
  - KeyEvent.KEY_PRESSED
  - KeyEvent.KEY_RELEASED
  - KeyEvent.KEY_TYPED

Получение информации

event.getCode()

event.getText()

event.getCharacter()

event.isShiftDown()

event.isControlDown()

event.isAltDown()

События мыши

- MouseEvent
  - MouseEvent.ANY
  - MouseEvent.MOUSE_PRESSED
  - MouseEvent.MOUSE_RELEASED
  - MouseEvent.MOUSE_CLICKED
  - MouseEvent.MOUSE_MOVED
  - MouseEvent.MOUSE_DRAGGED
  - MouseEvent.MOUSE_ENTERED
  - MouseEvent.MOUSE_EXITED

Прокрутка

- ScrollEvent
  - ScrollEvent.ANY
  - ScrollEvent.SCROLL
  - ScrollEvent.SCROLL_STARTED
  - ScrollEvent.SCROLL_FINISHED