Block223 Project

Team #22

Some added (optional features)

Admittedly, the project was hard as is... but we have some cool features nonetheless:

- Full color spectrum availability for block creation
- Showing a dynamically changing block color
- The ability to log out from any screen
- Easy access to all pages, at any time

Did you use any technology other than Umple, Java Swing, or Java 2D?

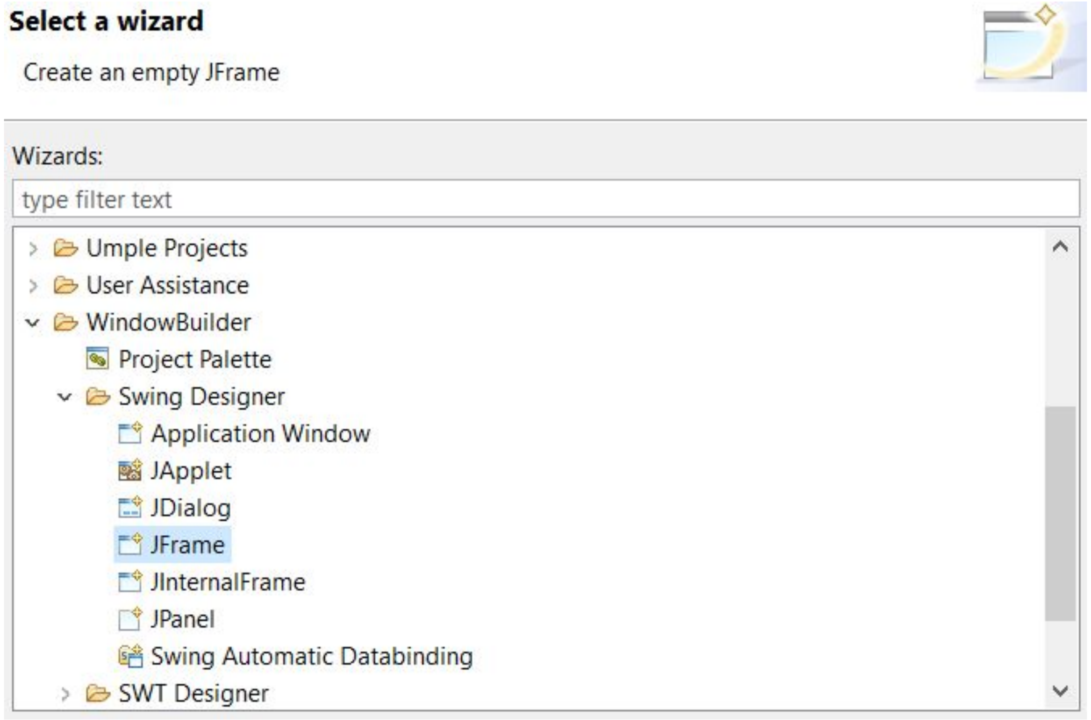
Java Swing Window Builder :

- A plugin which allowed for the creation of JFrames using a “drag and drop” system. The code would then be dynamically auto-generated for graphical objects such as JButtons , JTextFields , and password fields.
- Window Builder was not used for any playGame functionality, only for menu panels in view.

The setup for Window Builder

Can't edit an already created JFrame, have to create and name a new one from scratch.

If we want to add to an existing JFrame, code would have to be manually added into a JFrame builder which can cause many conflicts.



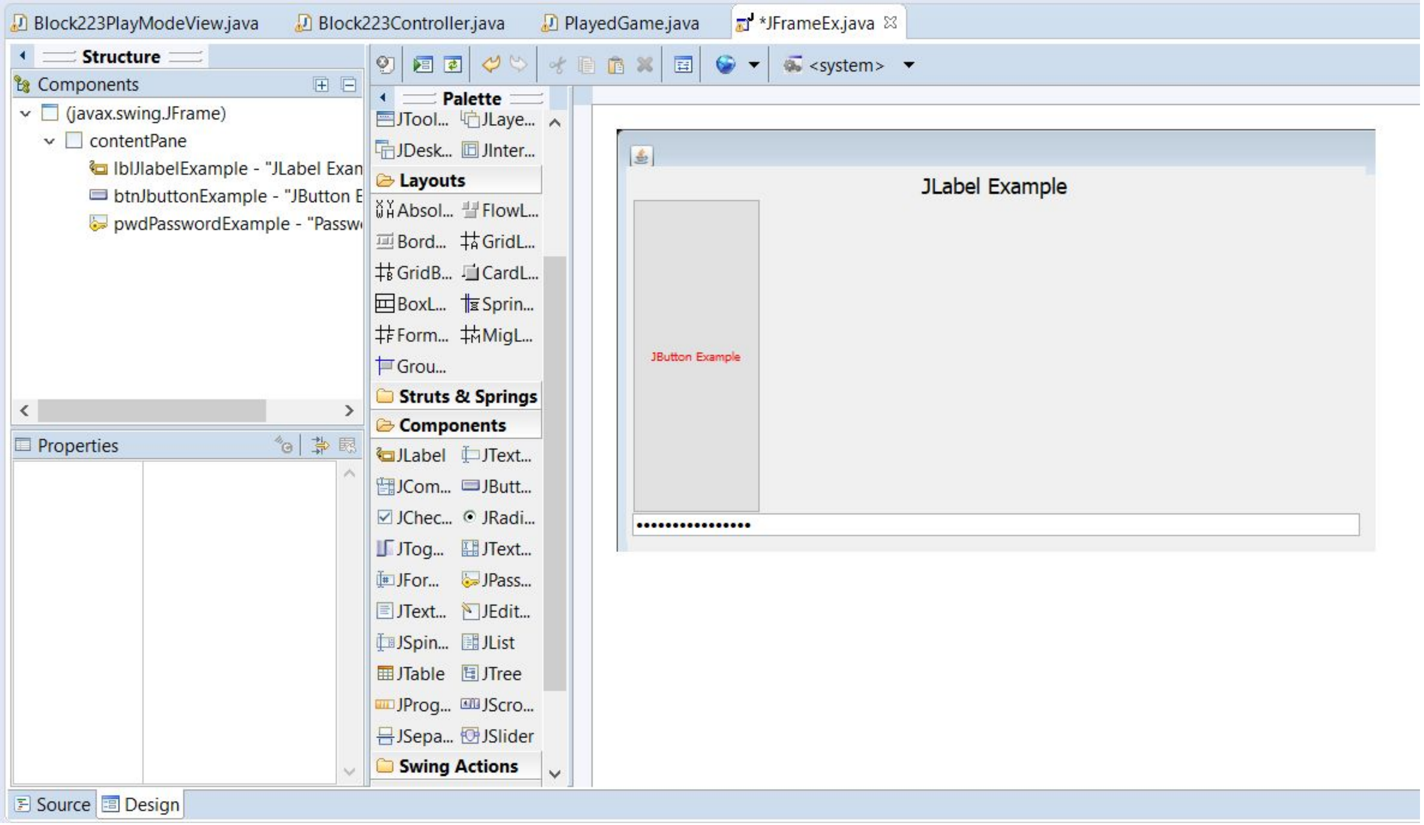
Select a wizard

Create an empty JFrame

Wizards:

type filter text

The UI design panel (drag and drop or manually edit)

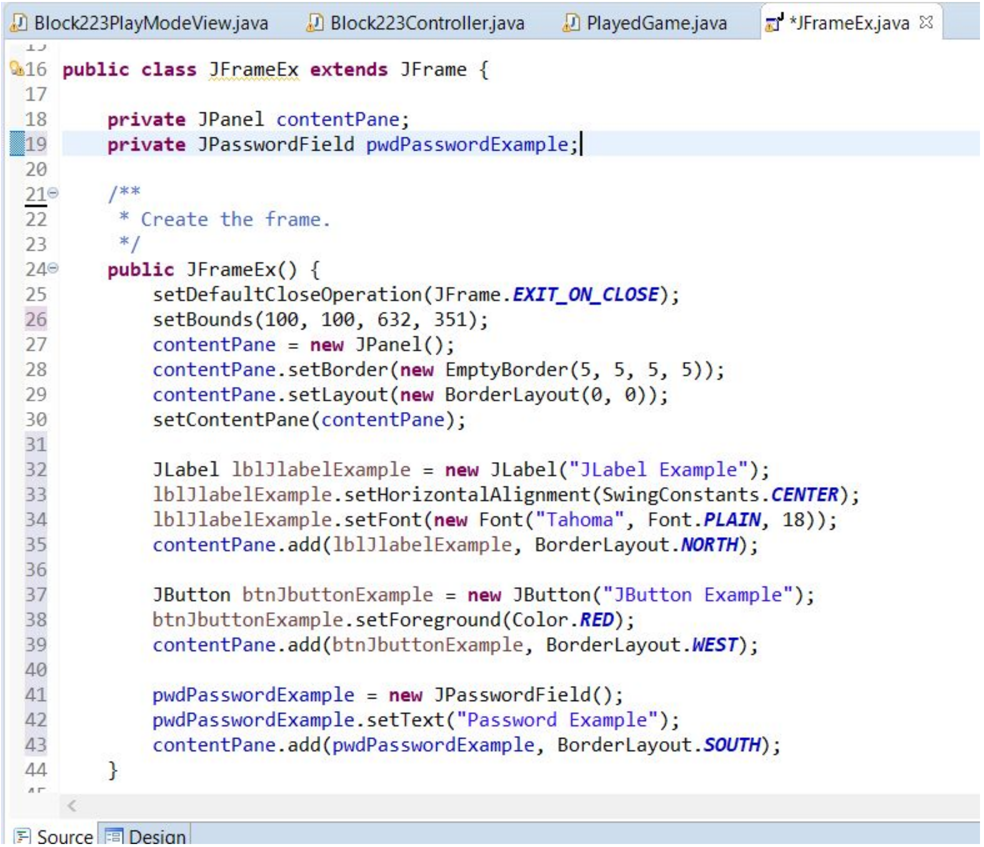


When changing from design to source window...

Code is automatically regenerated for additional features. Main() generated which would be commented out / worked around with application for MVC pattern).

Code that is changed manually stays the same.Action listeners would be added within this auto-generated code.

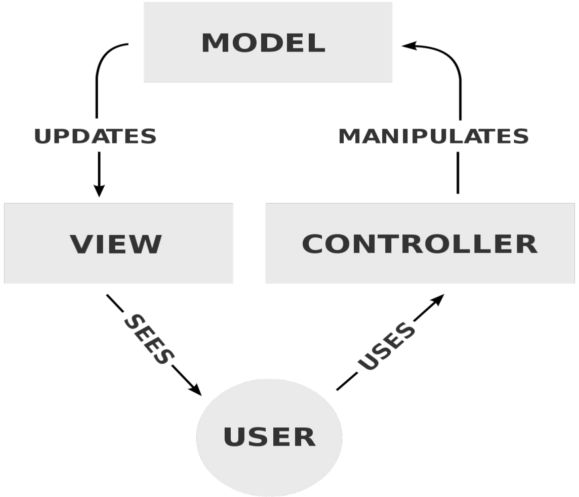
Similar to object oriented programming?



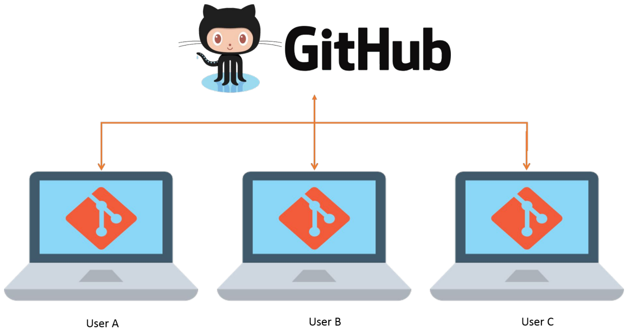
```java
public class JFrameEx extends JFrame {

    private JPanel contentPane;
    private JPasswordField pwdPasswordExample;

    /**
     * Create the frame.
     */
    public JFrameEx() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 632, 351);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.setLayout(new BorderLayout(0, 0));
        setContentPane(contentPane);

        JLabel lblJlabelExample = new JLabel("JLabel Example");
        lblJlabelExample.setHorizontalAlignment(SwingConstants.CENTER);
        lblJlabelExample.setFont(new Font("Tahoma", Font.PLAIN, 18));
        contentPane.add(lblJlabelExample, BorderLayout.NORTH);

        JButton btnJbuttonExample = new JButton("JButton Example");
        btnJbuttonExample.setForeground(Color.RED);
        contentPane.add(btnJbuttonExample, BorderLayout.WEST);

        pwdPasswordExample = new JPasswordField();
        pwdPasswordExample.setText("Password Example");
        contentPane.add(pwdPasswordExample, BorderLayout.SOUTH);
    }
```

Some lessons learned...

- The choice of patterns is very crucial, and MVC made things very easy
  - First went through drawing diagrams, and then coding
  - With these patterns, it was easy to decide where certain methods had to be called based on their functionality. These concepts almost ensured that the code could be self-organized.
- How to design projects on a larger scale.
  - Git and Github definitely helped



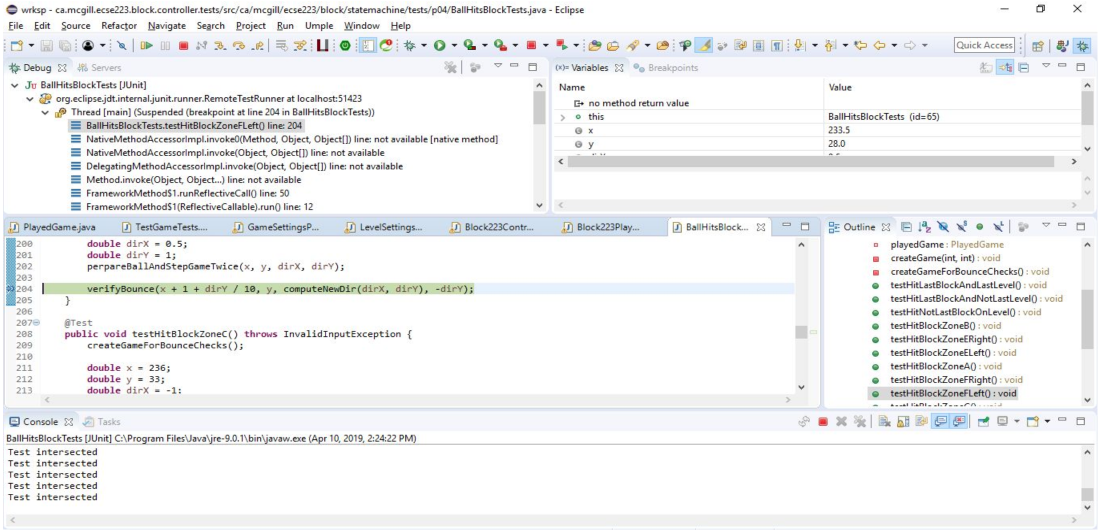
Source: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller



Source: https://www.edureka.co/blog/git-vs-github/

Some more lessons learned...

- The importance of debugging and testing without print statements.
    - Bugs found through the JUnit tests often involved the use of many different classes, which would certainly not have been found if only print statements were used.


wrksp - ca.mcgill.ecse223.block.controller.tests/src/ca/mcgill/ecse223/block/statemachine/tests/p04/BallHitsBlockTests.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Umple Window Help
Quick Access
Debug
Servers
Variables
Breakpoints
PlayedGame.java
TestGameTests....
GameSettingsP...
LevelSettings...
Block223Contr...
Block223Play...
BallHitsBlock...
Outline
Console
Tasks

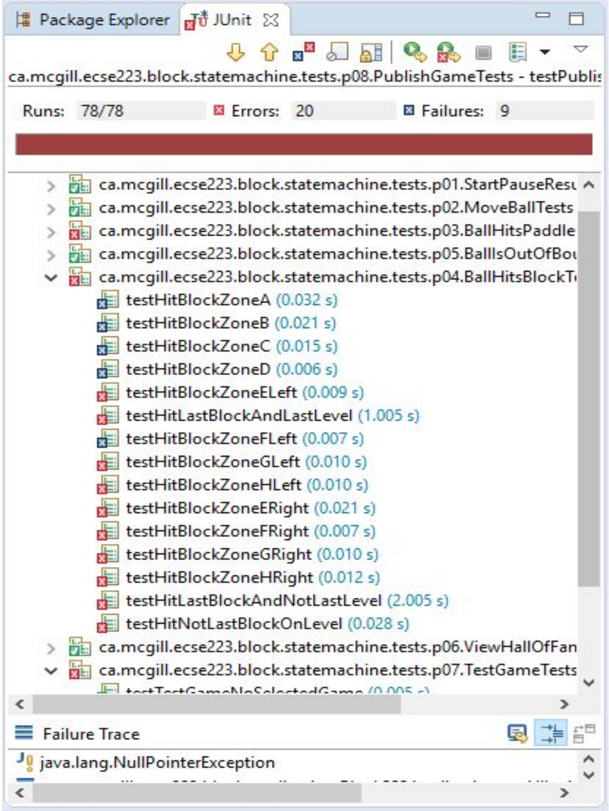
```
BallHitsBlockTests [JUnit]
  org.eclipse.jdt.internal.junit.runner.RemoteTestRunner at localhost:51423
    Thread [main] (Suspended (breakpoint at line 204 in BallHitsBlockTests))
      BallHitsBlockTests.testHitBlockZoneFLeft() line: 204
      NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native method]
      NativeMethodAccessorImpl.invoke(Object, Object[]) line: not available
      DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: not available
      Method.invoke(Object, Object...) line: not available
      FrameworkMethod$1.runReflectiveCall() line: 50
      FrameworkMethod$1(ReflectiveCallable).run() line: 12
```

Name	Value
no method return value	
this	BallHitsBlockTests (id=65)
x	233.5
y	28.0

```java
        double dirX = 0.5;
        double dirY = 1;
        perpareBallAndStepGameTwice(x, y, dirX, dirY);

        verifyBounce(x + 1 + dirY / 10, y, computeNewDir(dirX, dirY), -dirY);
    }

    @Test
    public void testHitBlockZoneC() throws InvalidInputException {
        createGameForBounceChecks();

        double x = 236;
        double y = 33;
        double dirX = -1:
```

```
playedGame : PlayedGame
createGame(int, int) : void
createGameForBounceChecks() : void
testHitLastBlockAndLastLevel() : void
testHitLastBlockAndNotLastLevel() : void
testHitNotLastBlockOnLevel() : void
testHitBlockZoneB() : void
testHitBlockZoneERight() : void
testHitBlockZoneELeft() : void
testHitBlockZoneA() : void
testHitBlockZoneFRight() : void
testHitBlockZoneFLeft() : void
```

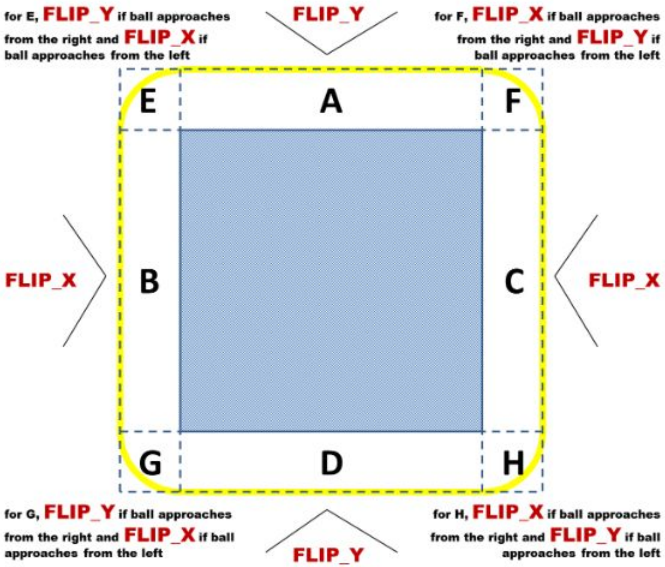
```
BallHitsBlockTests [JUnit] C:\Program Files\Java\jre-9.0.1\bin\javaw.exe (Apr 10, 2019, 2:24:22 PM)
Test intersected
Test intersected
Test intersected
Test intersected
Test intersected
```

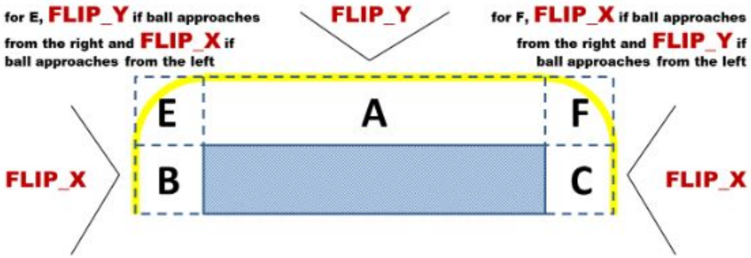


```
ca.mcgill.ecse223.block.statemachine.tests.p08.PublishGameTests - testPublis

Runs:  78/78     Errors:  20     Failures:  9

> ca.mcgill.ecse223.block.statemachine.tests.p01.StartPauseResu
> ca.mcgill.ecse223.block.statemachine.tests.p02.MoveBallTests
> ca.mcgill.ecse223.block.statemachine.tests.p03.BallHitsPaddle
> ca.mcgill.ecse223.block.statemachine.tests.p05.BallIsOutOfBou
v ca.mcgill.ecse223.block.statemachine.tests.p04.BallHitsBlockT
    testHitBlockZoneA (0.032 s)
    testHitBlockZoneB (0.021 s)
    testHitBlockZoneC (0.015 s)
    testHitBlockZoneD (0.006 s)
    testHitBlockZoneELeft (0.009 s)
    testHitLastBlockAndLastLevel (1.005 s)
    testHitBlockZoneFLeft (0.007 s)
    testHitBlockZoneGLeft (0.010 s)
    testHitBlockZoneHLeft (0.010 s)
    testHitBlockZoneERight (0.021 s)
    testHitBlockZoneFRight (0.007 s)
    testHitBlockZoneGRight (0.010 s)
    testHitBlockZoneHRight (0.012 s)
    testHitLastBlockAndNotLastLevel (2.005 s)
    testHitNotLastBlockOnLevel (0.028 s)
> ca.mcgill.ecse223.block.statemachine.tests.p06.ViewHallOfFan
v ca.mcgill.ecse223.block.statemachine.tests.p07.TestGameTests

java.lang.NullPointerException
```

A final lesson learned...

- The obvious distribution of features is not always the best.
  - Some features solve very similar problems.
  - E.g., hitting paddle and hitting block
  - Results in duplicated efforts


for E, FLIP_Y if ball approaches from the right and FLIP_X if ball approaches from the left
FLIP_Y
for F, FLIP_X if ball approaches from the right and FLIP_Y if ball approaches from the left
E
A
F
FLIP_X
B
C
FLIP_X
G
D
H
for G, FLIP_Y if ball approaches from the right and FLIP_X if ball approaches from the left
FLIP_Y
for H, FLIP_X if ball approaches from the right and FLIP_Y if ball approaches from the left


for E, FLIP_Y if ball approaches from the right and FLIP_X if ball approaches from the left
FLIP_Y
for F, FLIP_X if ball approaches from the right and FLIP_Y if ball approaches from the left
E
A
F
FLIP_X
B
C
FLIP_X

Thank you for your time!