MEMO

Date:5/30/2014

Subject: BlackJack

To: Virgil Bistriceanu

From: Geetha Sankineni

Introduction:

This project attempts to design and implement a Java application that allows the user to play the game of Blackjack.

Key Points:

- This project is designed applying the Object Oriented concepts of Java.
- GUI is implemented using SWING and AWT.
- Project is packaged using Java Web Start (JAWS).
- The system does not use any database.
- External resources and development tools used in the project are open-source.
- Implements almost all the major conventional rules applicable in a game of Blackjack (‘Split’ is not implemented).

Discussion:

Analysis:

The game of Blackjack is one of the most widely played casino banking games in the world (source: wikipedia.org). The requirements for this project are obtained from the information on Internet (http://www.pagat.com/banking/blackjack.html and Wikipedia). The system implements the object-oriented design. It is abstracted into several modules and specific functionality has been defined for each module.

Since the project is packaged using JAWS, the application can be easily downloaded by hitting the host url and run in local systems installed with Java. The application also checks for the latest version each time it is launched to keep it updated with latest changes.

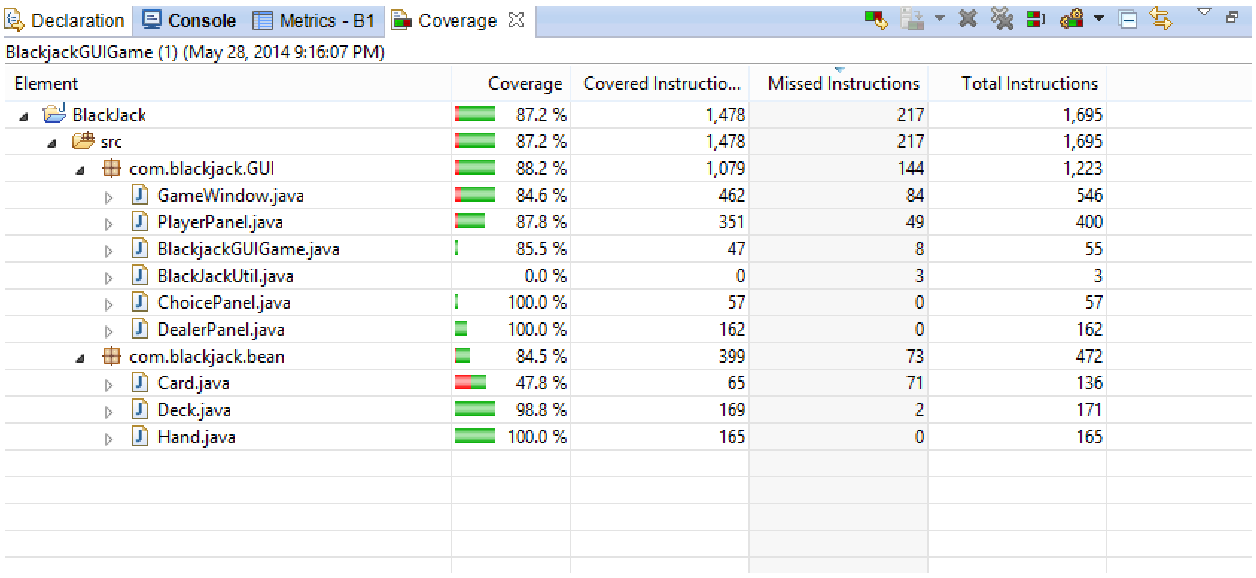
The main module starts the application and prepares the GUI with separate panels for dealer, player and choices using SWING. On reception of the player’s bet, cards are dealt both to the dealer and the player. Based on the choices made by the player, the game progresses to decide a winner based on standard BlackJack rules and pays out money. The system implements several rules like bet, hit, stand etc. to handle player’s choices in the game.

The player starts the game by placing bets from his available money. If the bet amount is more than or equal to the minimum bet, the player is dealt the first two cards of his hand. He then chooses either to hit or stand in order to get as closer to the BlackJack value (21) as possible. He might get busted in the due course. Once he decides to stand, the dealer's turn is started.

The Dealer has a pre-defined set of rules for hitting. He has to hit until his hand is greater than or equal to 17. He might get busted and the player wins automatically. If his hand has reached 17 or more, he can hit no more and stands for result.

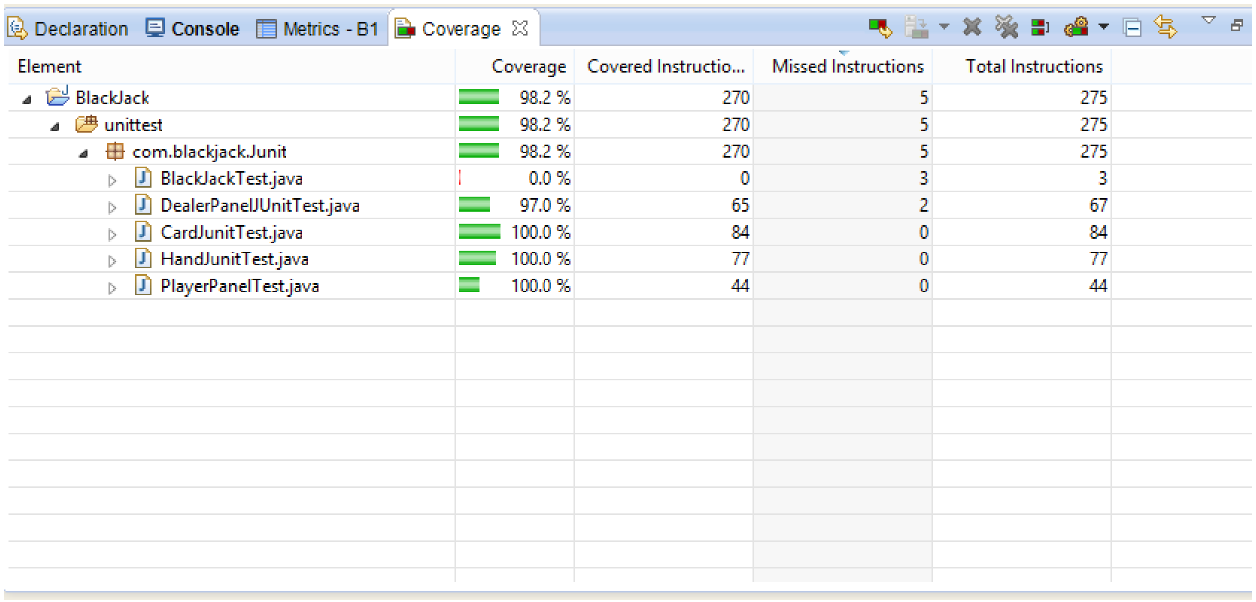
After the player and dealer have finished their turns, the system gets the best possible value of each hand and compares their values to decide a winner and pays out the bets. Tied values result in a Push and the bets are returned. The game restarts if the player runs out of money or the deck is fully dealt. The application exits if no bets are placed. All the resources and tools used in developing this project are open-source.

Unit test coverage report:



BlackjackGUIGame (1) (May 28, 2014 9:16:07 PM)

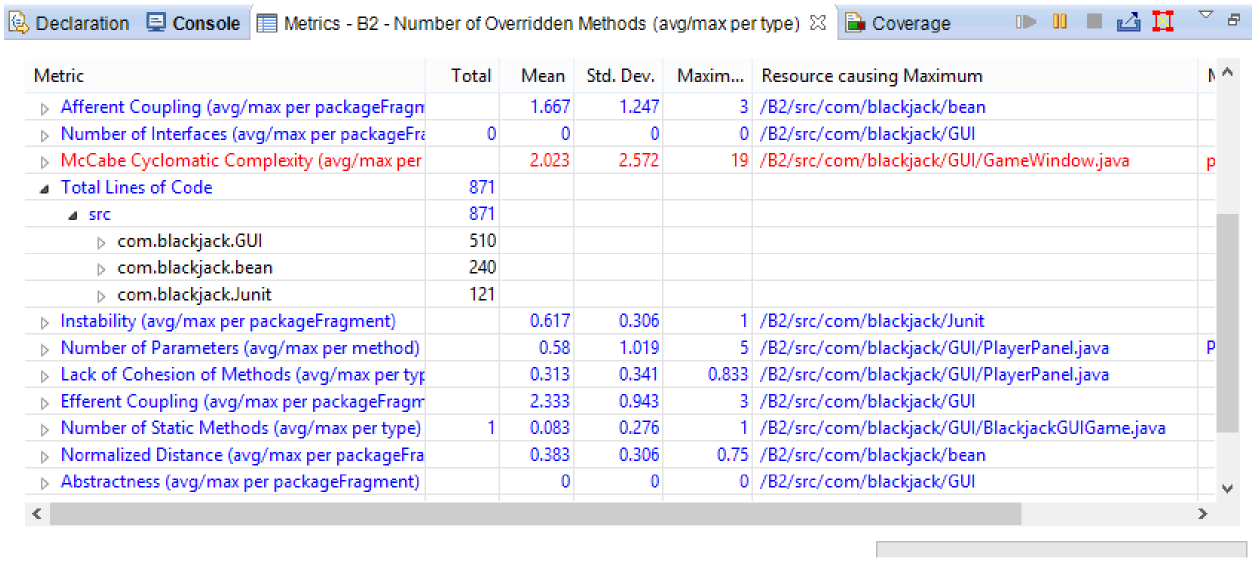
Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
BlackJack	87.2 %	1,478	217	1,695
src	87.2 %	1,478	217	1,695
com.blackjack.GUI	88.2 %	1,079	144	1,223
GameWindow.java	84.6 %	462	84	546
PlayerPanel.java	87.8 %	351	49	400
BlackjackGUIGame.java	85.5 %	47	8	55
BlackJackUtil.java	0.0 %	0	3	3
ChoicePanel.java	100.0 %	57	0	57
DealerPanel.java	100.0 %	162	0	162
com.blackjack.bean	84.5 %	399	73	472
Card.java	47.8 %	65	71	136
Deck.java	98.8 %	169	2	171
Hand.java	100.0 %	165	0	165



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
BlackJack	98.2 %	270	5	275
unittest	98.2 %	270	5	275
com.blackjack.Junit	98.2 %	270	5	275
BlackJackTest.java	0.0 %	0	3	3
DealerPanelJUnitTest.java	97.0 %	65	2	67
CardJunitTest.java	100.0 %	84	0	84
HandJunitTest.java	100.0 %	77	0	77
PlayerPanelTest.java	100.0 %	44	0	44

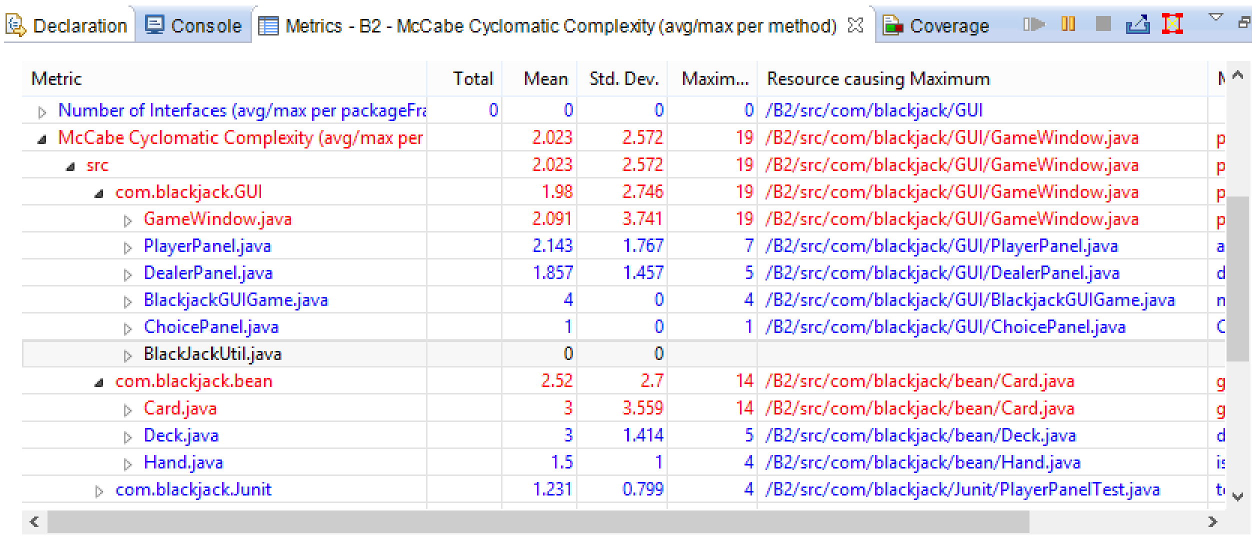
The unit test coverage for all the unit test cases is above 85%

Lines Of Code:


Declaration
Console
Metrics - B2 - Number of Overridden Methods (avg/max per type)
Coverage

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	N
Afferent Coupling (avg/max per packageFragn		1.667	1.247	3	/B2/src/com/blackjack/bean	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/B2/src/com/blackjack/GUI	
McCabe Cyclomatic Complexity (avg/max per		2.023	2.572	19	/B2/src/com/blackjack/GUI/GameWindow.java	p
Total Lines of Code	871					
src	871					
com.blackjack.GUI	510					
com.blackjack.bean	240					
com.blackjack.Junit	121					
Instability (avg/max per packageFragment)		0.617	0.306	1	/B2/src/com/blackjack/Junit	
Number of Parameters (avg/max per method)		0.58	1.019	5	/B2/src/com/blackjack/GUI/PlayerPanel.java	P
Lack of Cohesion of Methods (avg/max per typ		0.313	0.341	0.833	/B2/src/com/blackjack/GUI/PlayerPanel.java	
Efferent Coupling (avg/max per packageFragn		2.333	0.943	3	/B2/src/com/blackjack/GUI	
Number of Static Methods (avg/max per type)	1	0.083	0.276	1	/B2/src/com/blackjack/GUI/BlackjackGUIGame.java	
Normalized Distance (avg/max per packageFra		0.383	0.306	0.75	/B2/src/com/blackjack/bean	
Abstractness (avg/max per packageFragment)		0	0	0	/B2/src/com/blackjack/GUI	

Cyclomatic Complexity:


Declaration
Console
Metrics - B2 - McCabe Cyclomatic Complexity (avg/max per method)
Coverage

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	M
Number of Interfaces (avg/max per packageFra	0	0	0	0	/B2/src/com/blackjack/GUI	
McCabe Cyclomatic Complexity (avg/max per		2.023	2.572	19	/B2/src/com/blackjack/GUI/GameWindow.java	p
src		2.023	2.572	19	/B2/src/com/blackjack/GUI/GameWindow.java	p
com.blackjack.GUI		1.98	2.746	19	/B2/src/com/blackjack/GUI/GameWindow.java	p
GameWindow.java		2.091	3.741	19	/B2/src/com/blackjack/GUI/GameWindow.java	p
PlayerPanel.java		2.143	1.767	7	/B2/src/com/blackjack/GUI/PlayerPanel.java	a
DealerPanel.java		1.857	1.457	5	/B2/src/com/blackjack/GUI/DealerPanel.java	d
BlackjackGUIGame.java		4	0	4	/B2/src/com/blackjack/GUI/BlackjackGUIGame.java	n
ChoicePanel.java		1	0	1	/B2/src/com/blackjack/GUI/ChoicePanel.java	C
BlackJackUtil.java		0	0			
com.blackjack.bean		2.52	2.7	14	/B2/src/com/blackjack/bean/Card.java	g
Card.java		3	3.559	14	/B2/src/com/blackjack/bean/Card.java	g
Deck.java		3	1.414	5	/B2/src/com/blackjack/bean/Deck.java	d
Hand.java		1.5	1	4	/B2/src/com/blackjack/bean/Hand.java	is
com.blackjack.Junit		1.231	0.799	4	/B2/src/com/blackjack/Junit/PlayerPanelTest.java	t

Time spent on project:

Number of hours needed to get the code working is 270 hours
Number of hours spent preparing submission is 30 hours

Challenges faced:

- The game is played with different set of rules in different parts of the world. A prominent variant was researched to implement consistent rules.
- Some effort was required to develop a good look and feel interface using Swing.
- The advantages of packaging using JAWS were understood while developing multiple versions of the application.

Recommendations:

i. The current system does not implement Split, double down rules of Blackjack. This can be worked upon as an enhancement.
ii. It is also desirable to support multiple players in the application.

Conclusion:

Blackjack game has been successfully implemented using the specified technologies.