**Inline singleton refactoring**

The Console class in this case is the singleton, which instantiates the responses in a game of blackjack. The ScenarioTest and Blackjack classes call methods from the singleton to determine the current state of play. Rather than doing this, the methods can be referenced within the classes themselves instead of from the singleton. This is achieved below.

public class ScenarioTest extends TestCase...

public void testDealerStandsWhenPlayerBusts() {

**~~Console.setPlayerResponse(new TestAlwaysHitResponse());~~**

int[] deck = { 10, 9, 7, 2, 6 };

Blackjack blackjack = new Blackjack(deck);

**blackjack.setPlayerResponse(new TestAlwaysHitResponse());**

blackjack.play();

assertTrue("dealer wins", blackjack.didDealerWin());

assertTrue("player loses", !blackjack.didPlayerWin());

assertEquals("dealer total", 11, blackjack.getDealerTotal());

assertEquals("player total", 23, blackjack.getPlayerTotal());

}

public class Blackjack...

public void play() {

deal();

writeln(player.getHandAsString());

writeln(dealer.getHandAsStringWithFirstCardDown());

HitStayResponse hitStayResponse;

do {

write("H)it or S)tay: ");

hitStayResponse = **~~Console.~~**obtainHitStayResponse(input);

write(hitStayResponse.toString());

if (hitStayResponse.shouldHit()) {

dealCardTo(player);

writeln(player.getHandAsString());

}

}

while (canPlayerHit(hitStayResponse));

// ...

}

The Console class can now be deleted from the program