



SpiderLogic  
10000 Innovation Drive  
Milwaukee, WI 53226

Phone: 414.290.8015  
E-mail: info@spiderlogic.com

Where architects code and coders architect!



## Sheepshead ... The New "Techie" Game

By Chris Peterson

I have had a series of posts on the card game Sheepshead. As you may know, we play the game at work during lunch because of the interesting concept of power vs. points.

If you are not a Sheepshead player, you should know the following:

First, Sheepshead is a German card game which is very native to Wisconsin. In fact, it is incredibly difficult to find anyone without Wisconsin roots who plays the game.

Second, Sheepshead is very difficult to learn to play. The basic strategy is to take tricks and get points, similar to other games like Hearts or Bridge. However, the power ranking and point values of cards is very confusing and takes a while to understand.

As we struggle to teach and include new players, we have been trying to come up with a way to make the game easier for beginners to understand, and

allow advanced players a way to keep the game fresh.

So here is the SpiderLogic take on Sheepshead.

### Specialized Sheepshead Cards

It would be great if there were specialized Sheepshead cards. (you only use 32 cards) Instead of the pictures of royalty and number patterns on the cards, we would suggest just a big picture of the suit, for example, a big heart for hearts, a big club for clubs, a big spade for spades, and a big T for trump. This would cut down on the confusion on what is trump and what is not. In addition, along the top of the cards the point value (11, 10, 4, 3, 2, 0) should be printed. Down the side you can put the trump order. (Q, J, A, 10, K, 9, 8, 7, clubs, spades, hearts, diamonds)

### Computerized Help

It's hard not to put a software spin on our approach. With the addition of a bar code to each card, the following approach

could be implemented. Each player would have a card scanner that would scan the card being tossed. A computer program could beep a warning if the player was not going to follow suit and make a misplay. You could also program it for other things like optimal play so if you are a beginner, it could help you play better.

This same system could also track points for the hand. This would save time, allowing for more hands played. I know what you are saying "good Sheepshead players should already know the point count." Well true, but you still need to do a confirming count at the end of the game.

### Stats Keeping

I envision the mother of all stats keeping. Scanning cards would allow for some great stats to be collected. Not just basic things like how many times I pick and win. I am talking like: How often does the called suit walk? How many times did I pick and who was my partner? Do I have a better win percentage in three handed? Does one player tend to pick and pull down others?

Of course all this would be displayed on the web for all to see. These stats would be fresh in real time. Talk about bragging rights.

### Online Play ... Forget it!

I have tried to play Sheepshead online and really it is not that fun. First, it is really hard to get a table going if you are not a rated player. I have been playing Sheepshead for 13 years but because I never have played online, I have no rating and can't get on a table.

Checkout Chris' blog:  
[chriscpeterson-blog.blogspot.com](http://chriscpeterson-blog.blogspot.com)



# Quarterly Spin



Volume 1, Issue 3  
Summer 2007

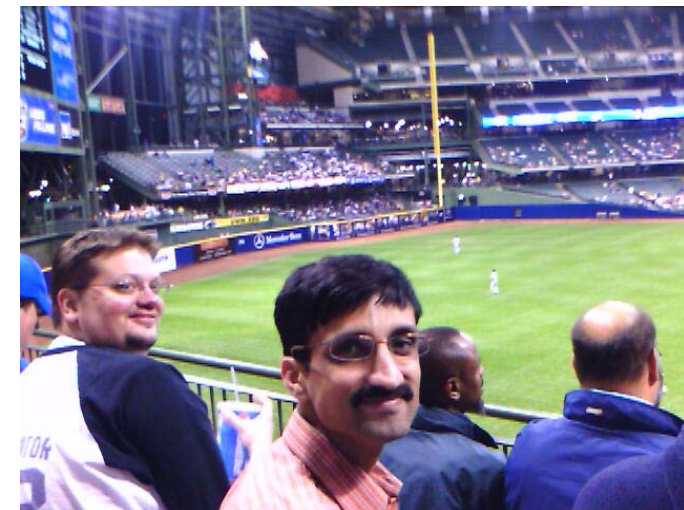
## Brewers Welcome Kartik of India

By Dave Buettner

The resurgence of the Milwaukee Brewers this year has caused incredible excitement throughout the state of Wisconsin. As passionate as we may think we are about "America's Pastime," our enthusiasm may take a backseat to that of the average "cricket fan" in India.

Recently, Kartik Rindani, a Senior Architect from our Pune, India office visited us in Milwaukee and was given a crash course on baseball. It is impossible for the avid cricket fan not to draw comparisons between the "gentlemanly" sport of cricket and the "barbarous" game of baseball.

Sure ... a baseball game of 3 hours must look "lightning fast" to the cricket fan who is used to a game lasting 3 days.



Despite the inherent differences, Kartik was a fast learner. He grasped the terminology quickly and became quite proficient at understanding some of the subtle nuances.

With training complete, it was time for Kartik to experience his first Brewers game. First, of course, was the obligatory tailgate party. Many of the Milwaukee Spiders and their families were also in attendance.

Continued on page 2



## On Singletons

By Geoff Lane

The [Singleton Pattern](#) is one of the most widely used patterns from the [Gang of Four \(GoF\) Design Patterns Book](#). One of the reasons that it's so widely used, I think, is because it's also very easy to understand. The basic idea is that you control the creation of an object so that it will only ever have one instance. This is achieved by having a private constructor and a static instance that is accessed through a static method on the Class.

Continued on page 3

### Singleton Example:

```
public class FirstSingleton {
    private static final FirstSingleton
instance = new FirstSingleton();

    public static FirstSingleton in-
stance() {
        return instance;
    }

    private FirstSingleton() {
    }
}
```

Check out Geoff's blog at  
[zorched.net](http://zorched.net)



### Inside this issue:

<i>On Singletons</i>	1
<i>Brewers Welcome Kartik of India</i>	1
<i>Sheepshead the New Techie Game</i>	4





## Brewers ... Kartik continued from pg. 1

Once inside Miller Park, we climbed the ramp and headed to our seats in the left field bleachers. It was a beautiful night for baseball and the roof was open. Chris Capuano the left hander was pitching that night.

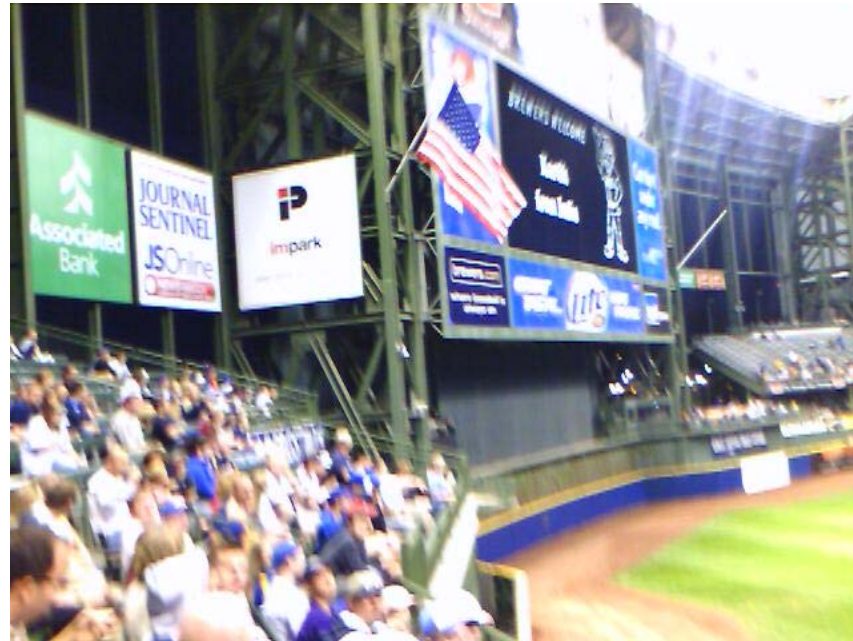
Kartik asked many questions and was intrigued by the concept of pitching from the stretch and holding the runner on first base. Foul balls were also a new concept, as in cricket, there is no such thing as a foul ball. You can hit it in any direction.

As the game continued, we noticed a pattern. Every time the Brewers came out to warm up for the next inning, Geoff Jenkins, the Brewer's Left Fielder, would throw warm up tosses with the ball girl located on the left field line. When the warm up time ended, Jenkins would throw the ball up into the bleachers. After the first few innings, we got the hang of it and joined the rest of the left field "bleacher creatures" and screamed for the ball.

Adding to the magic of the night, our own Geoff Lane snagged the ball from Jenkins with an incredible one handed grab. Even more incredible was that he made the catch without spilling his beer. Geoff to Geoff connection complete.



Geoff Lane showing off the baseball



Brewers Scoreboard welcoming Kartik

As is the tradition, the Brewers use their large scoreboard to welcome groups, celebrities, and other dignitaries. Around the sixth inning, the Brewers appropriately acknowledged Kartik's first visit to Miller park with a very large ...

### Brewers Welcome Kartik from India

on the score board. Our section became very excited, and we all stood up and pointed to Kartik so everyone in the left field bleachers would know who he was. The entire bleacher section cheered for Kartik and wished him well.

Spider Brian Kapellusch quipped, "Hey Kartik, don't get too excited about the scoreboard greeting. There could be lots of other Kartiks here tonight."

Next, the Sausage Race. Clearly, this event is well loved in Brewer Fan circles but is a difficult concept to explain to our India colleagues. The race is comprised of competing giant sausages representing different nationalities. They race around the field to the delight of the crowd.

Noticeably absent was a vegetarian entry. I think the Brewers should consider adding some type of tofu or other soy product to the race.

The game continued and Capuano pitched a brilliant game. He struck out a season high nine batters and scattered seven hits. For the record, this was his last winning effort of the season ... so far. Following this game, he has been in a major slump. Could it be that his good luck charm has returned to India?

Baseball is a very superstitious sport.

Brewers won 3-0.

To learn more about the Brewers, you may want to check out brewerfan.net.

A website for Brewer fans run by our own Brian Kapellusch.



## On Singletons continued from pg. 1

### Reasons to Avoid the Singleton Pattern

#### Practical Limitations

There are some practical limitations of the Singleton pattern though. With modern VM based languages like Java, there can be multiple classloaders in a given application. A Singleton then is no longer a Singleton as there will be an instance-per-classloader. So if you can't enforce an actual single instance, why use the Singleton Pattern? You might answer that the classloader case is a rare one and that in a "normal" application you're not going to run into that. And you would be right. But I still think that you should avoid the Singleton.

#### Multiple Concerns

Fundamentally a Class in Object Oriented Programming should have one and only one responsibility. The [Single Responsibility Principle](#) often leads to good design and the benefits of a highly cohesive system such as being easily understood and being easier to change and maintain over time.

The Singleton, by its very definition, has multiple responsibilities. The primary responsibility defined in the pattern is controlling constructions of Objects of this Class. Of course, from an application point of view, this is pretty weak. What you really want from a class is a useful service. So if you have a Singleton Data Access class, which is more important? The Singularity of the instance or the ability to get at data in your data store? I really hope you picked the latter.

There are a number of ways that you can control access to the construction of a class that do not rely on the Singleton pattern. A Factory or a Container, for example, can be used to get at an instance in such a way that the calling code doesn't care about the implementation of the construction of the class. Whether a new instance is created or a single instance returned should be immaterial to the user of that instance. This also does a good job of dealing with our Single Responsibility concern raised earlier. Now the enforcement of a single instance is handled by another class whose sole responsibility is that singleton enforcement.

#### Testability

One consideration that must be made when you are designing software is: How are you going to test the code? If no consideration is given, then often the code will be very difficult to test and quality will suffer.

```
public class SecondSingleton {
    private static final SecondSingleton instance = new SecondSingleton();
    private SomeDependency myDependency;

    public static SecondSingleton instance() {
        return instance;
    }

    private SecondSingleton() {
        myDependency = new SomeDependency();
    }
}
```

The example of a Singleton above is relatively simple. But what happens if you introduce a dependency to your Singleton? Now you have a tight-coupling between two classes. If the dependent class does something like access the database or call a web service or itself has a bunch of dependencies then all of a sudden the code becomes very hard to test.

I've written previously about [Mock Objects](#) for testing. Mock Objects allow you to create dummy implementations of dependencies to make testing a Class much easier. In the Singleton example here, you can see that there would be no way to inject a Mock instance into the SecondSingleton class.

```
public class NotSingleton {
    private SomeDependency myDependency;

    private NotSingleton(SomeDependency depends) {
        myDependency = depends;
    }

    public class Factory {
        private NotSingleton notsingleInstance = new NotSingleton(
            (SomeDependency) lookupType(SomeDependency.class));

        public NotSingleton getNotSingleton() {
            return notsingleInstance;
        }

        /**
         * Get an instance for a given Interface
         * @param interface The interface for which you want the instance
         */
        public Object lookupType(Class clazz) {
            // ...
        }
    }
}
```

#### Factories or Containers to Enforce Single Instances

One of the best ways to access dependencies in a class is through [Constructor Based Dependency Injection](#). This can be done through a container like [Spring](#) or [Pico Container](#).

You can, of course, do this yourself as well. If you are using a Factory for object creation, then the singleton enforcement and the dependency injection can happen in the same class (whose sole responsibility is to manage the construction of objects).

With this structure you can now independently test all of your classes without worrying about the interactions of dependent objects. You also have not structured the code in such a way that having multiple instances should cause any problems, so your code won't break if the classloader case ever comes up.