Sharing the state between the steps in cucumber using PicoContainer. this means that we must be able to share state between steps.

The glue between Gherkin and the system under test are implemented as regular java methods and implemented in regular java classes. The steps are global in the sense that every step in the same package or sub-package relative to the runner will be found and executed. This allows us to define one step in one class and another step in another class.

When there was only one class, an instance variable or two was probably enough. Now you need to solve the problem with a shared state between the two or more classes with steps.

**Sharing the state between different classes – Dependency injection.**

A native solution in java could be share a state using a class with static fields, this will work. It is unfortunately very for information to leak from one scenario to another. Static field are not cleared while the JVM is running. To clear them, we would either have to reset them manually or restart the JVM, both ways are cumbersome.

The solution in java is to use dependency injection. That is, inject a common object in each class with steps. An object that is recreated every time a new scenario is executed.

Dependency injection can be done in many ways. A simple solution is to inject dependeinces through the constructor (also called Constructor Dependency Injection).

Cucumber-JVM support many different dependency injection frameworks and PicoContainer is one of the frameworks.

sharing state between steps that depends on the result of an earlier step in the scenario. Sharing state can be done in different ways.

Singleton:

The purpose of the Singleton class is to control object creation limiting the number of objects to only one.

The singleton allows only one entry point to create the new instance of the class.

Since there is only is one Singleton instance, any instance fields of a Singleton will occur only once per class, just like static field. Singletons are often useful where you have to control the resources, such as database connections or sockets.

Restrict the instantiation of a class and ensures that only one instance of the class exists in the Java Virtual Machine.

**Inheritance concept**:

**Overriding**: defining a method in the child class with same name and same signature in the parent class. [Note: **child class should extend the parent class**]

**Why can’t we override static methods?**

No, static means common for all the objects in the hierarchy. When we want to override then there is no point to declare the method as a static in the parent class.

**Why we cannot override constructor?** [ constructor in java is special method in java]

No, the constructor name of the child class should be same as the child class name then, how can you call the parent class constructor. Constructor is a special method belongs to the specific class only.

static: // memory management.

* Static variables: declared inside class and out a method
* Static methods: method is declared static
* Static blocks: using static keyword.

Static variable, methods and static block are executed during .class file loading.

When we execute a java class file (java Classname)

* Loads corresponding .class byte code into memory.
* After loading, it calls main method for execution.

[note: .class is loaded only once].

**Instance block depends on the object creation.**

JVM memory areas:

* Method area:
* Heap area:
* Stack area:
* PC registers
* Native method:

Local variables:

* Declared inside method, constructor and blocks.
* Scope of the local variable specific to method, constructor and blocks.
* Memory is allocated only when the method is invoked, the memory is released with is completed.

Instance Variables:

* Declared inside the side class and outside the methods.
* Scope of the instance variable is inside the class all methods, constructors and blocks.
* Memory is allocated when the object is created, and memory destroyed when is destroyed.
* Stored in heap memory.
* Can be accessed directly if same area (instance area), if other area (static) using the object.

Static Variables:

* Declared inside the class and outside the methods with static modifier.
* Scope of the instance variable is inside the class all methods, constructors and blocks.
* Memory is allocated when the .class is loading and destroyed .class file
* Stored in non- heap memory.
* Can be assessed by using classname.

Areas of java:

* Instance area:
* Static area:

Heap memory: objects occupy memory.

Stack memory: references pointing to object in heap memory.

**static and non-static variables (also called as instance variables**:

Variable is named memory location.

Any thing which is common, then we have to go for static variable, free access, no restrictions.

Static belongs to a class.

They are accessed using Classname.variableName outside the class.

Every method of a class can access a static variable including Constructor.

this(): this() keyword is used to call constructor.

In constructor overloading, if we want to call same class constructor from another constructor.

this: this keyword used to call the variable and methods in the same class.

super(): super() keyword is used to call super class constructor.

Is mainly used to class parent class constructor.

Super: super keyword is used to call variable and methods from the super class.

String concept:

When we create string object it will create memory in SCP (String constant pool) memory.

It will always check for previous object with same content before creating object. If available, it will not create object but the reference variable will be pointing to existing object.

SCP will not allow duplicate values.

Once we create string object, we cannot perform modification are not allowed. Because String is immutable class.

When we create string object, it will create memory in Heap memory.

It will not check for previous object with same content, but will create a new object.

Heap area will allow duplicate values.

Once we create stringBuffer object, we can perform modification. Because Stringbuffermutable class.