

**اصول شبیه سازی**

**گزارش از روند حل مسائل**

**استاد درس**

جناب آقای دکتر احمدی

**تهیه کنندگان**

**فاطمه نیک خواه - 8828096**

**علی لشگری - 8758717**

**محمدرضا تأثیری - 8812076**

*[This page intentionally left blank]*

[](https://github.com/taesiri/Simulation)**[](https://github.com/taesiri/Simulation)**

Fork Source Code on Github

<https://github.com/taesiri/Simulation>

We do love Open Source, Full Source code and project documentation is available on github.com/taesiri/Simulation.

*Taesiri*: The source code is published under my Github account just because I was the only person had active Github account at publishing date.

**فهرست مطالب**

**مختصر اطلاعات فنی در مورد پیاده سازی پروژها5**

**مساله 16**

**نمودار های جریان7**

**مساله 28**

*Class Diagram***11**

**نمودار های جریان12**

**اعداد تصادفی در** C#**17**

**اسامی تصادفی19**

**تایم لاین!21**

**ضمیمه 22**

*A Brief Technical Document on Projects Implementation*

**مختصر اطلاعات فنی در مورد پیاده سازی پروژها**

Total Line of Codes: 1349 (Problem 1 with 467 and Problem 2 with 882) – Calculated with Code Metrics

Programming Languages Used: C#, IronPython, IronRuby

Application User interface entirely written in WPF using C#

Runtime Platform: .Net Framework 4 (Scripting features needs host has IronPython and/or IronRuby Installed)

زبان برنامه نویسی اصلی، زبان C# می باشد. تعداد خطوط نوشته شده در حدود 1500 خط برای هر دو مساله می باشد.

**مساله 1**

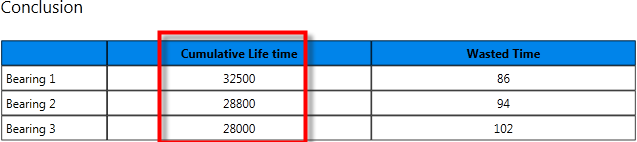
مساله پایایی مربوط به ماشین فرز دارای 3 برینگ را برای هر دو روش پیشنهادی؛ برای 20 خرابی یا تعویض از هر برینگ شبیه سازی نمایید. نمودارهای جریان مربوطه را نیز برای هر دو روش ترسیم نمایید.

*(سوال اشاره ای به مقایسه دو روش نکرده)*

*(در شبیه سازی این مساله، کارگر برای تعمییر دستگاه به تعداد کافی در اختیار داریم و صفی تشکیل نمی شود)*

**مقایسه دو روش**

با تکرار آزمایش به تعداد زیاد، روش دوم را روش بهینه معرفی می کنیم. مشاهده شده است که زمان خواب دستگاه کاسته شده است.



روش اول

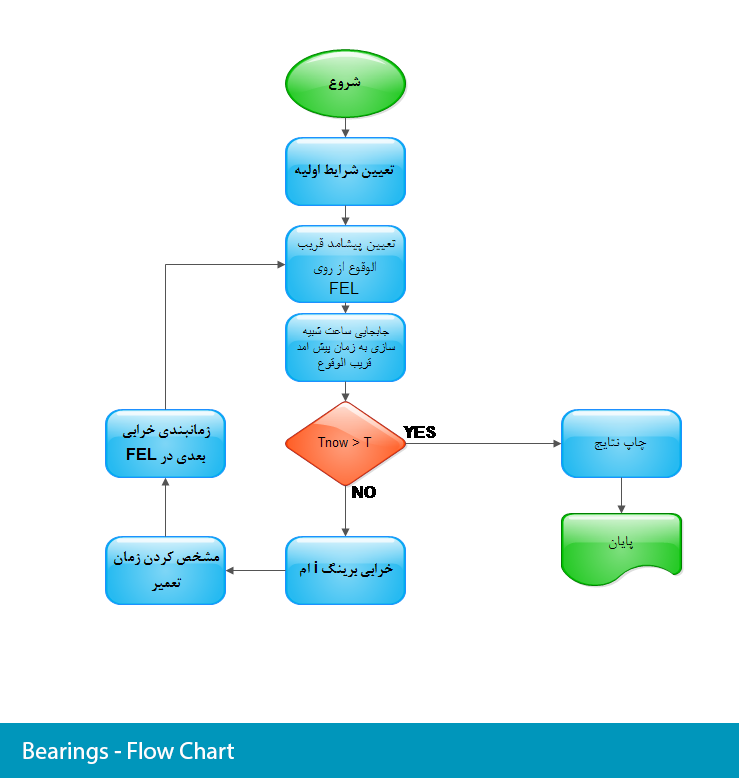


روش دوم

برای مشاهده نمونه گزارش های شبیه سازی به ضمیمه گزارش مراجعه کنید.

*Flow Chart*

**تذکر** : در شبیه سازی این مساله، برای بهبود سرعت (Performance) از روش FEL استفاده نشده؛ گرچه نمودارهای جریان بر اساس FEL گزارش داده شده اند.

****

**مساله 2**

*Special thanks to Google Goggle!*

Able, Baker, and Charlie are three carhops at the Sonic Drive-In (service at the speed of Sound). Cars arrive every 5 +- 5 minutes. The carhops service customers at the rate of one 10+-6 minutes. However the customers prefer Able over Baker and Baker over Charlie. If the carhop of choice is busy, the customers choose the ﬁrst available carhop.

Simulate the system for 1000 service completions. Estimate Able’s, Baker’s and Charlie’s Utilization (percentage of time busy).

Also, repeat the case where there are no preferences between carhops. Compare the results with the previous case. Which approach would be better?

Source:

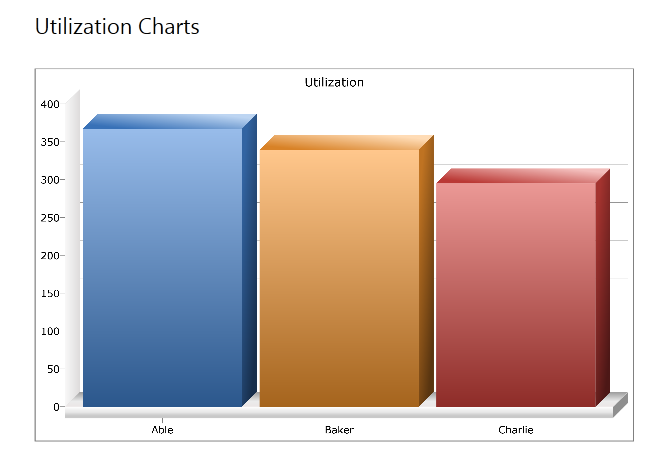
Exercise problems - Ugrad.cs.ubc.ca

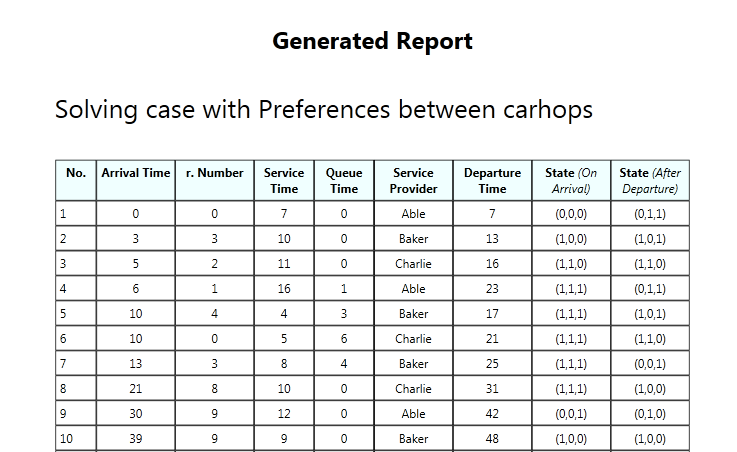
www.ugrad.cs.ubc.ca/~cs405/ch4excercises.pdf

**مقایسه دو روش**

با تکرار آزمایش به تعداد زیاد، در روش اول، Able بیشترین زمان خدمت دهی و بیشترین تعداد سرویس به مشتری را به خود اختصاص داده است. بعد از او Baker و Charlie به ترتیب قرار می گیرند.

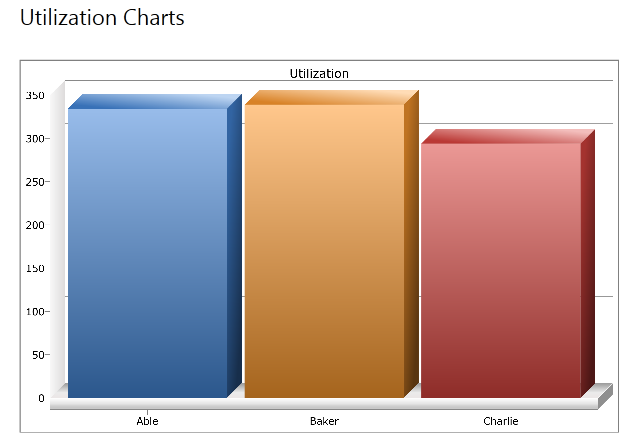
نمونه ای از خروجی برنامه در حالت اول

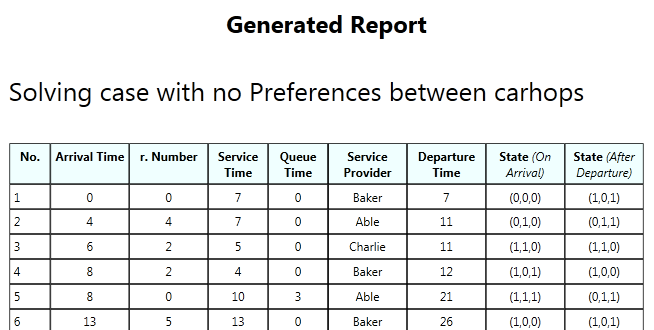




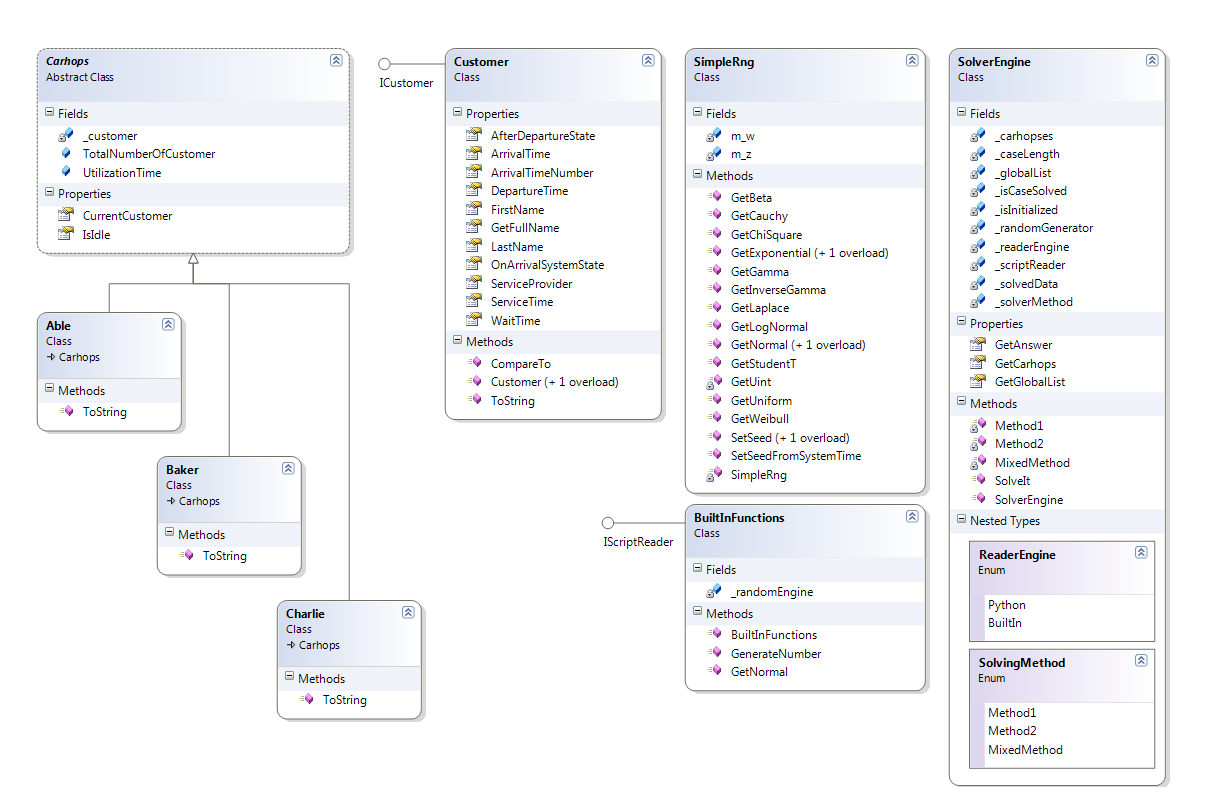
در شبیه سازی به روش دوم مشاهده می شود این میزان مشارکت سه خدمت دهنده به طور متوسط یکی است

نمونه ای از خروجی در حالت دوم

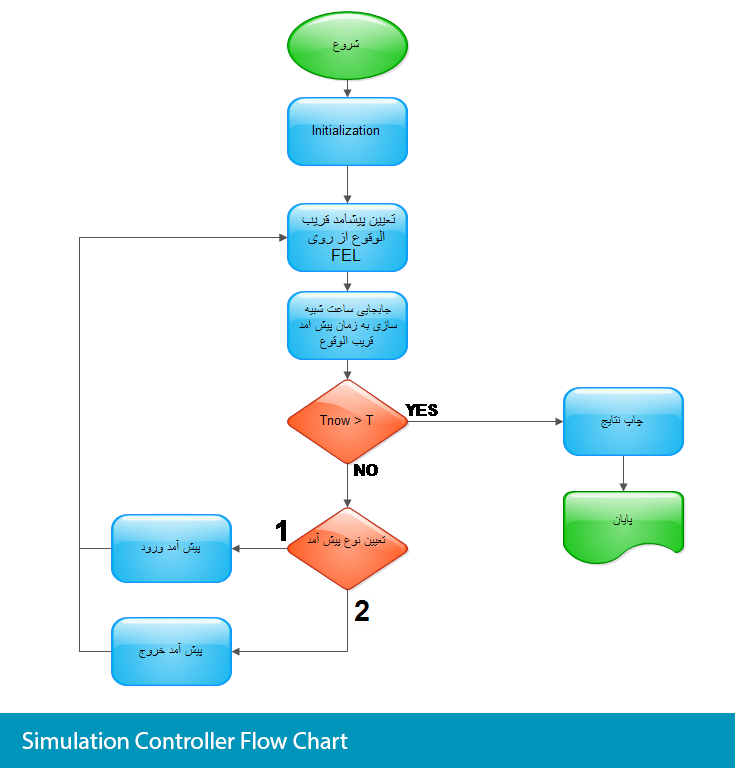


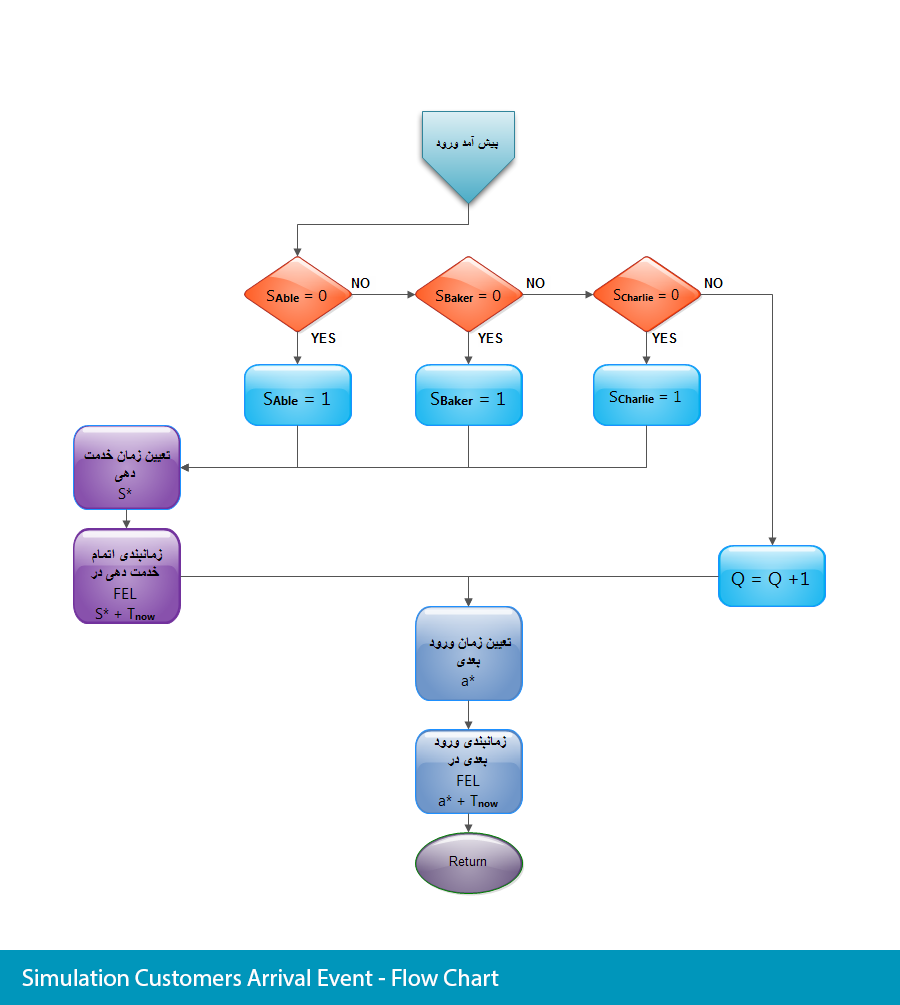


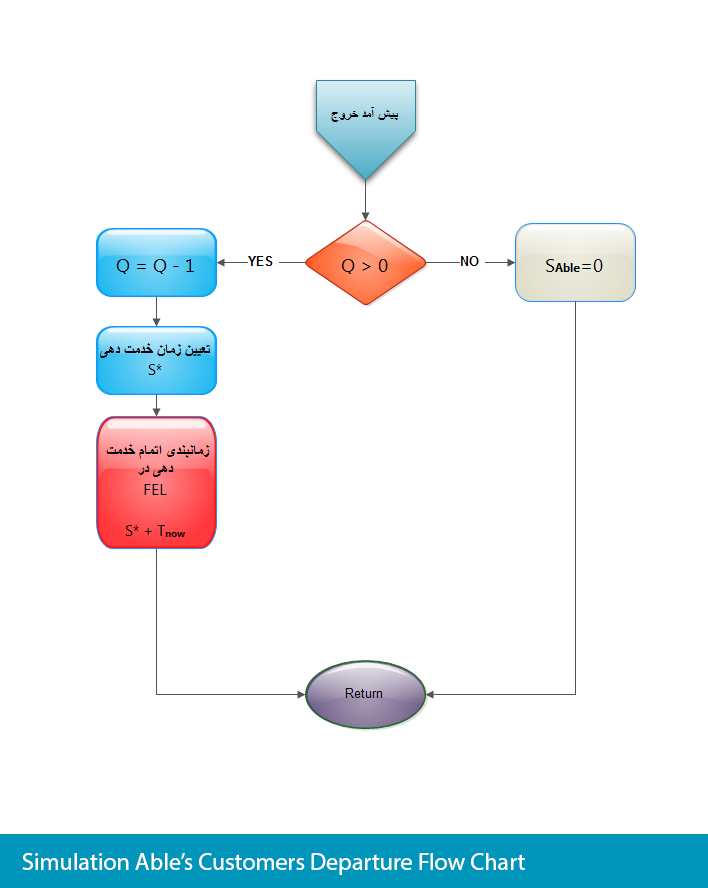
*Class Diagram*

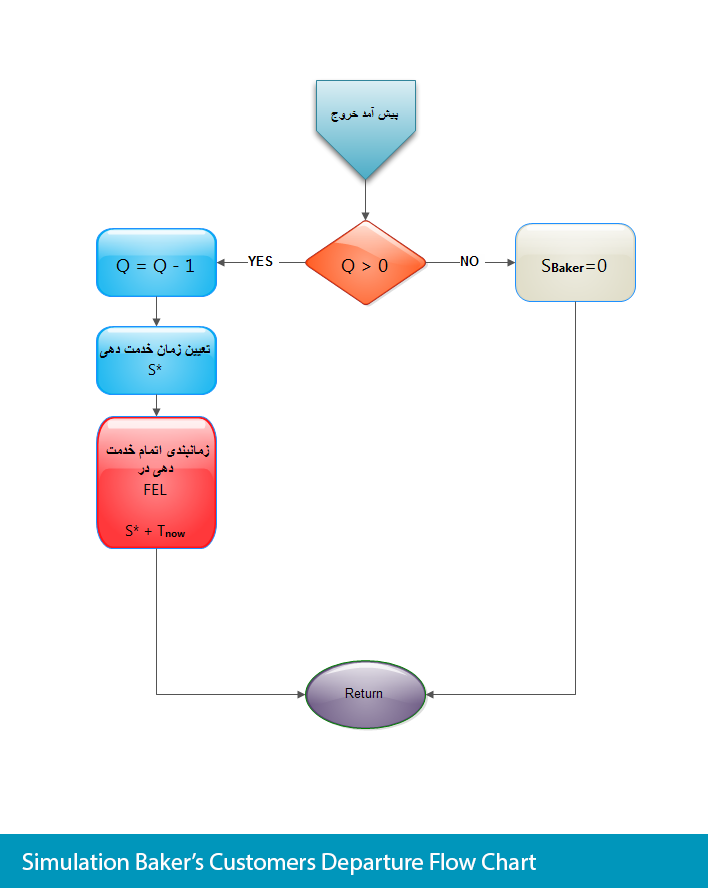
**

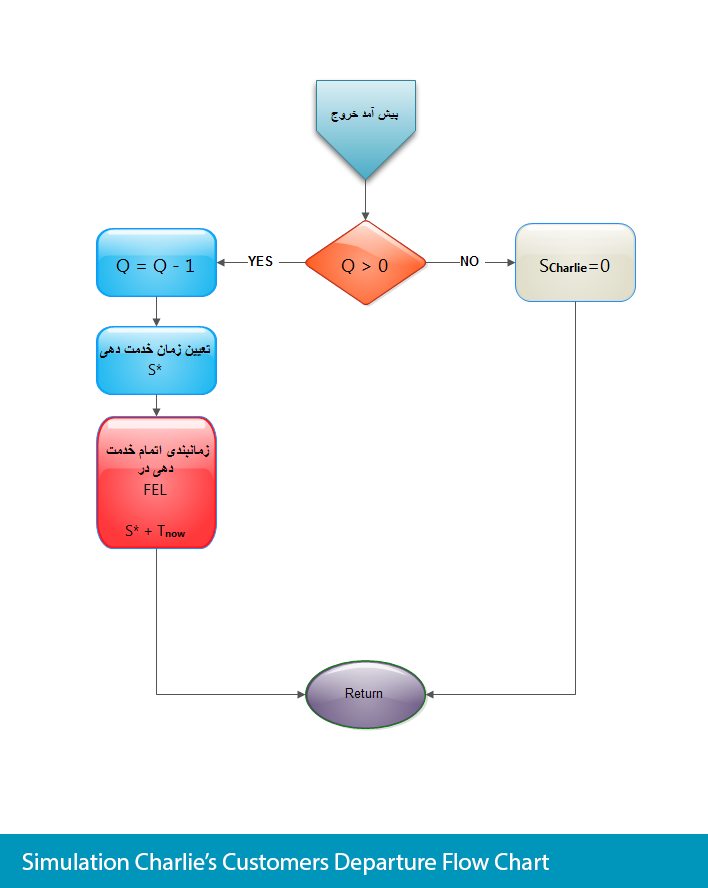
*Flow Charts*











**randn**

Normally distributed pseudorandom numbers

**اعداد تصادفی)** **توزیع نرمال اعداد شبه تصادفی( در C#**

همانطور که می دانید؛ در کتابخانه های زبان C#، تابع randn وجود ندارد. برای رفع این کاستی روش های گوناگونی وجود داشت، استفاده از کتابخانه های بسیار قدرتمند و گسترده در زمینه محاسبات عددی که به زبان C# پیاده سازی شده، و یا استفاده از توابع نرم افزار Matlab در زبان C#. ولی روشی که ما برگزیدیم؛ پیاده سازی یک الگوریتم برای تولید اعداد تصادفی با خاصیت مورد نظر بود.

**روش 1Box – Muller**

در حال حاضر، برنامه برای تولید اعداد شبه تصادفی از این الگوریتم استفاده می نماید. اعداد تولیدی با میانگین 0 و واریانس 1 هستند.

*اعداد تصادفی با میانگین 10 و انحراف معیار 5 تولید کنید*

var Mean = 10;

   var Deviation = 5;

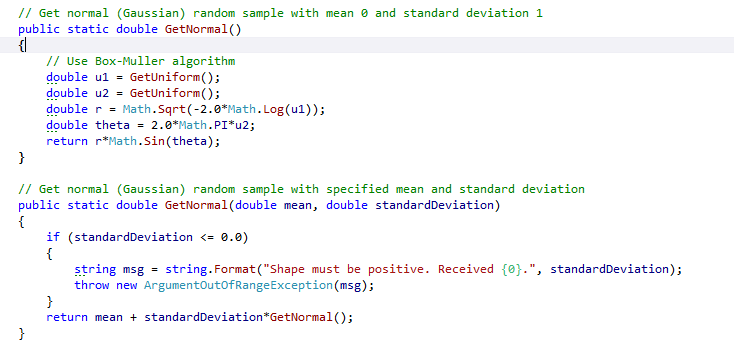
   return Mean + (Deviation \* GetNormal());

برای بدست آوردن مقادیر در بازه های دلخواه، از الگوی زیر پیروی می کنیم:

*در پروژه های بعدی تیم علاقه بسیاری دارد از الگوریتم Ziggurat برای تولید اعداد تصادفی استفاده نمایید. این الگوریتم بسیار سریع تر از الگوریتم فعلی می باشد.*

*کد مربوط به تولید اعداد تصادفی (نرمال) توسط John D. Cook پیاده سازی شده و به صورت متن باز در اختیار عموم قرار داده شده است. به دلیل احترام به جامعه متن باز اسم ایشان در برنامه ذکر شده است.*

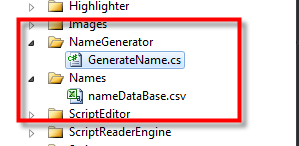
1. *در ضمیمه این گزارش، روش کار این الگوریتم به صورت کامل شرح داده شده است.*

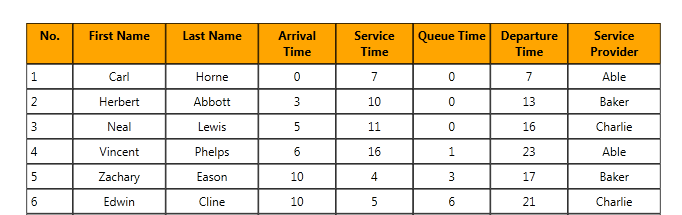


**اسامی تصادفی!**

در طراحی شبیه سازی مسئله دوم، گروه دست به ابتکاری جالب زده است و برای هر یک از مشتریان، یک نام به صورت تصادفی تولید می کند. این کار صرفا جهت بالا بردن کارایی نرم افزار شبیه ساز است.

برای تولید اسامی تصادفی، نرم افزار از یک بانک اطلاعاتی در فورمت CSV استفاده می کند. این بانک شامل 10.000 نام و نام خانوادگی می باشد.



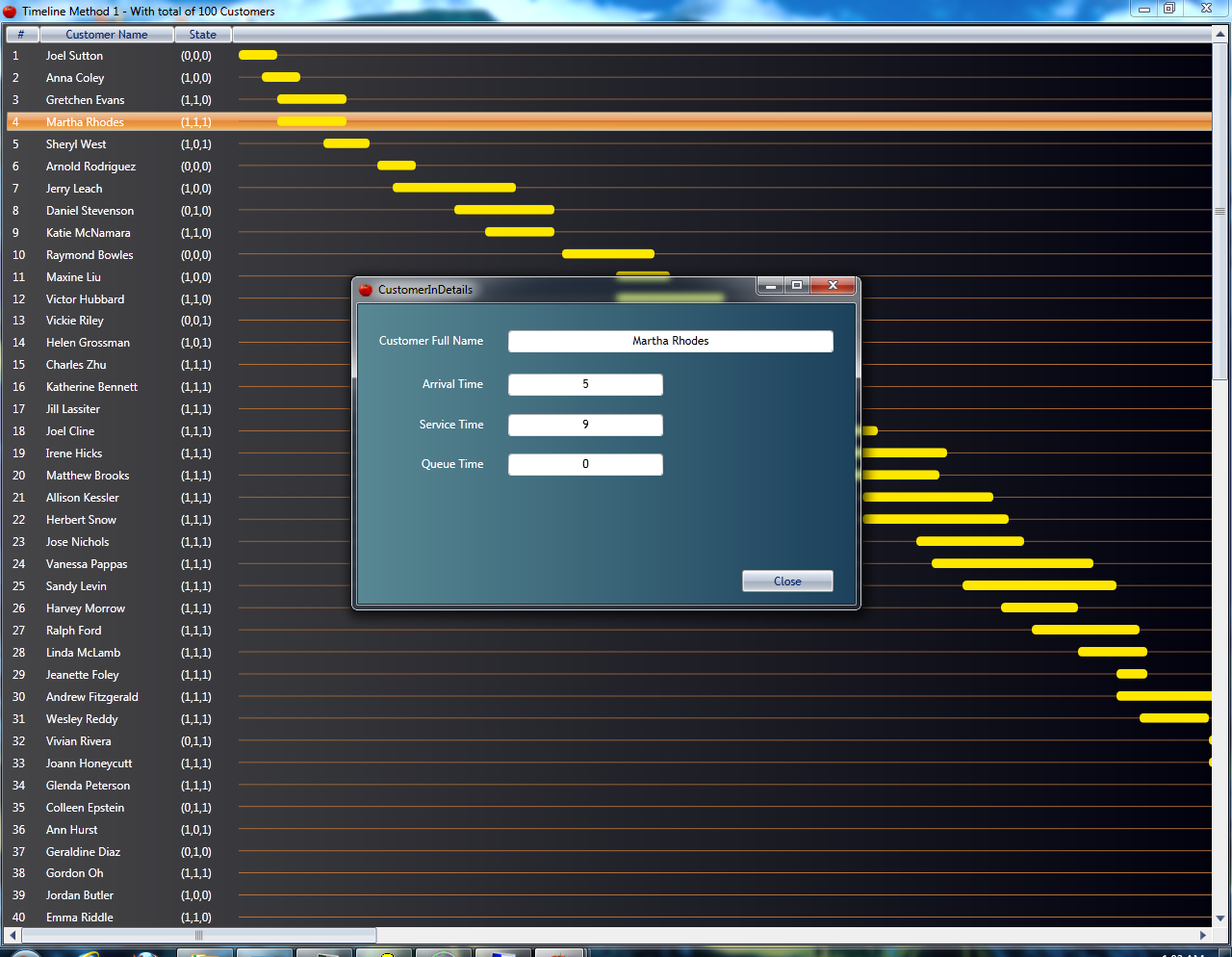


در صفحه ی بعد، کلاس مربوط به تولید اسامی تصادفی را می توانید مشاهده کنید.

*Timeline!*

تایم لاین!

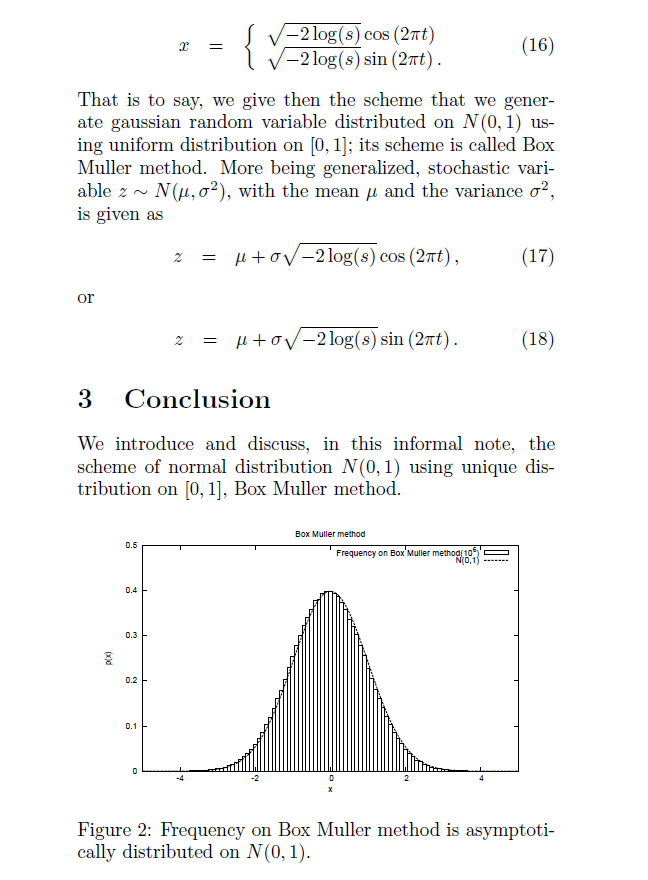
در مراحل پایانی پیاده سازی پروژه؛ ایده ای به ذهن گروه خطور کرد! نشان دادن زمان شبیه سازی به صورت گرافیکی!



**ضمیمه 1**

*الگوریتم Box-Muller*

boxmuller.pdf



boxmuller.pdf