

به نام خدا

ارزیابی کارایی سیستم‌های کامپیوتری

گزارش کار پروژه PECS
استاد عبداللهی ازگمی

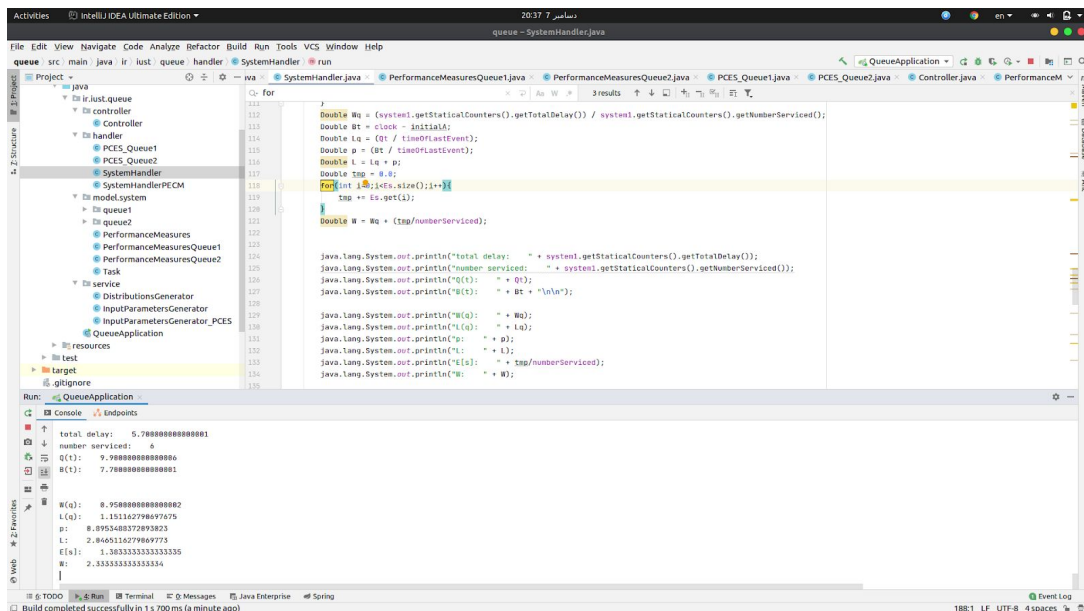
غزاله بختیاری آزاد
99723175

بررسی کلی عملکرد پروژه:

ابتدا در نظر بگیرید که پروژه دارای تنها یک صف است. در ابتدا پارامترهای ورودی شامل زمان‌های ورود و زمان‌های سرویس مقداردهی اولیه شده و بصورت لیست به یک تابع به نام `initializeSystemParameters` داده می‌شود که این تابع تمامی مقدارهای لازم شامل `System states` Statical counters, ... را صفر کرده و سیستم را آماده پذیرش تسک می‌کند. سپس تابع `run()` اجرا می‌شود که در هر `clock` بر اساس نوع رخداد تابع‌های مربوطه شامل `arrivalEventRaised()` و `exitEventRaised()` صدا زده شده و وضعیت سیستم به روز می‌شود که در هر مرحله پارامترهای لازم برای ارزیابی کارایی مقداردهی شده و در انتها با انجام محاسبات تمامی معیارهای کارایی بدست می‌آید.

اثبات درستی عملکرد پروژه:

برای اثبات درستی عملکرد کد، پارامترهای ورودی با مثالی که در اسلایدهای درس موجود است مقداردهی شدند و همانطور که در تصویر مشاهده می‌کنید تمامی معیارهای کارایی به درستی ارزیابی و نمایش داده شده‌اند:



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
queue - SystemHandler.java
src: main java ir.iust.queue.handler SystemHandler run
Project: java
  ir.iust.queue
    Controller
    handler
      PCES_Queue1
      PCES_Queue2
      SystemHandler
    model.system
      queue1
      queue2
    PerformanceMeasures
      PerformanceMeasuresQueue1
      PerformanceMeasuresQueue2
    Task
    service
      DistributionsGenerator
      InputParametersGenerator
      InputParametersGenerator_PCES
      QueueApplication
  resources
  test
  target
  gitignore

122 for
123 {
124     Double Wq = (system1.getStatisticalCounters().getTotalDelay()) / system1.getStatisticalCounters().getNumberServiced();
125     Double Bt = clock - initialAt;
126     Double Lq = (Qt / timeOfLastEvent);
127     Double p = (Bt / timeOfLastEvent);
128     Double L = Lq + p;
129     Double tmp = 0.0;
130     for(int i=0; i<ls.size(); i++)
131     {
132         tmp += ls.get(i);
133     }
134     Double W = Wq + (tmp/numberServiced);
135
136     java.lang.System.out.println("total delay: " + system1.getStatisticalCounters().getTotalDelay());
137     java.lang.System.out.println("number serviced: " + system1.getStatisticalCounters().getNumberServiced());
138     java.lang.System.out.println("Q(t): " + Qt);
139     java.lang.System.out.println("B(t): " + Bt + "\n\n");
140
141     java.lang.System.out.println("W(q): " + Wq);
142     java.lang.System.out.println("L(q): " + Lq);
143     java.lang.System.out.println("p: " + p);
144     java.lang.System.out.println("L: " + L);
145     java.lang.System.out.println("E[s]: " + tmp/numberServiced);
146     java.lang.System.out.println("W: " + W);
147 }

Run QueueApplication
Console Endpoints
total delay: 5.788888888888889
number serviced: 6
Q(t): 9.988888888888886
B(t): 7.788888888888889
W(q): 0.9588888888888882
L(q): 1.151162798697675
p: 8.895540037285825
L: 2.8465116279869773
E[s]: 1.3833333333333335
W: 2.3333333333333334

Build completed successfully in 1 s 700 ms (a minute ago)
```

تابع‌های توزیع:

برای این پروژه به دو نوع تابع توزیع (تابع توزیع نمایی با نرخ 1 و 2 - تابع توزیع نرمال (0.0 - 0.5) احتیاج داریم که کد این توزیع‌ها در قسمت زیر قابل مشاهده است:

- تابع توزیع نمایی با نرخ λ :

```
public double generateExponential(double lambda) {  
    if (lambda == 0.0)  
        return 0.0;  
    double randomNumber;  
    randomNumber = random.nextDouble();  
    return -1 / (lambda * Math.log(randomNumber));  
}
```

- تابع توزیع یکنواخت پیوسته (a, b) :

```
public double generateUniform(double a, double b) {  
    return (a + (b - a) * random.nextDouble());  
}
```

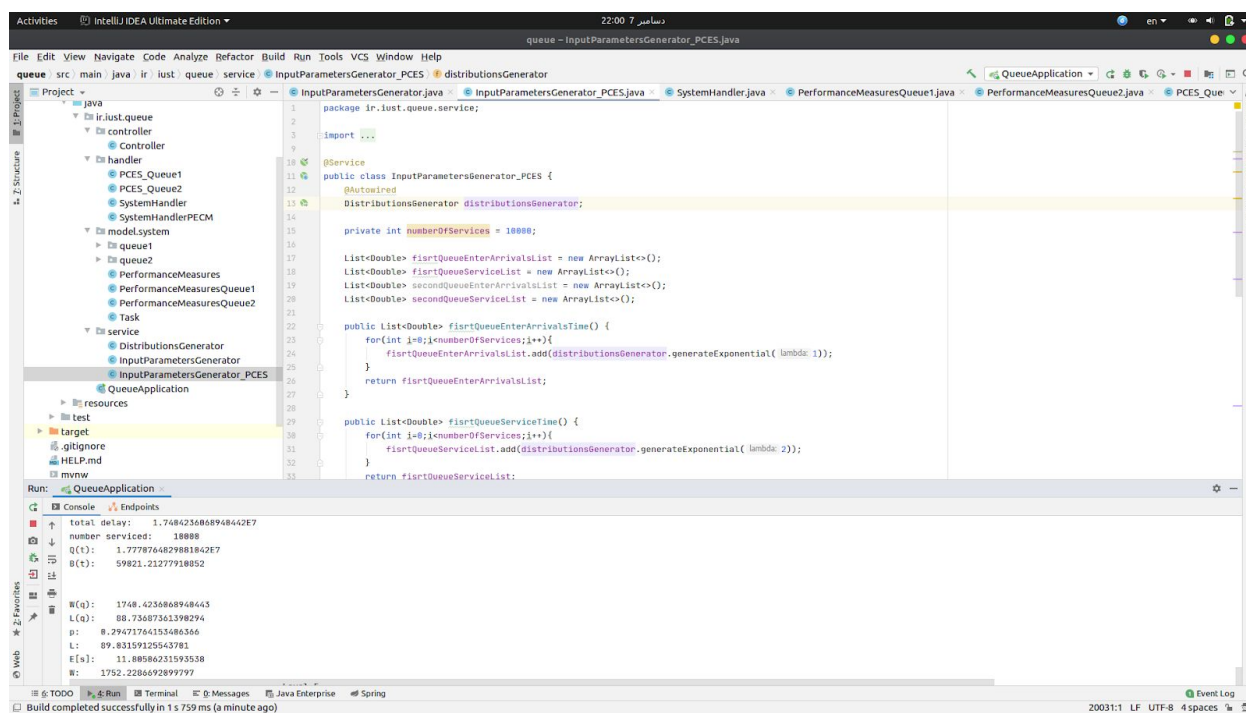
بررسی عملکرد پروژه با توجه به دو صف موجود در صورت سوال:

برای اینکار از روشی که در قسمت‌های بالا توضیح داده شد استفاده می‌کنیم و تسک‌ها را از صف اول عبور می‌دهیم حال با توجه به p مورد نظر انتخاب می‌کنیم که هر تسک پس از خروج از صف اول باید از سیستم خارج شود یا وارد صف دوم گردد برای اینکار برای هر تسک یک عدد رندوم بین 0 و 1 تولید می‌کنیم اگر این عدد رندوم از p تعیین شده کمتر باشد تسک مورد نظر کامل شده و از سیستم خارج می‌شود و اگر بزرگتر باشد تسک مورد نظر باید وارد صف دوم شود و پس از خروج از صف دوم از سیستم خارج می‌شود.

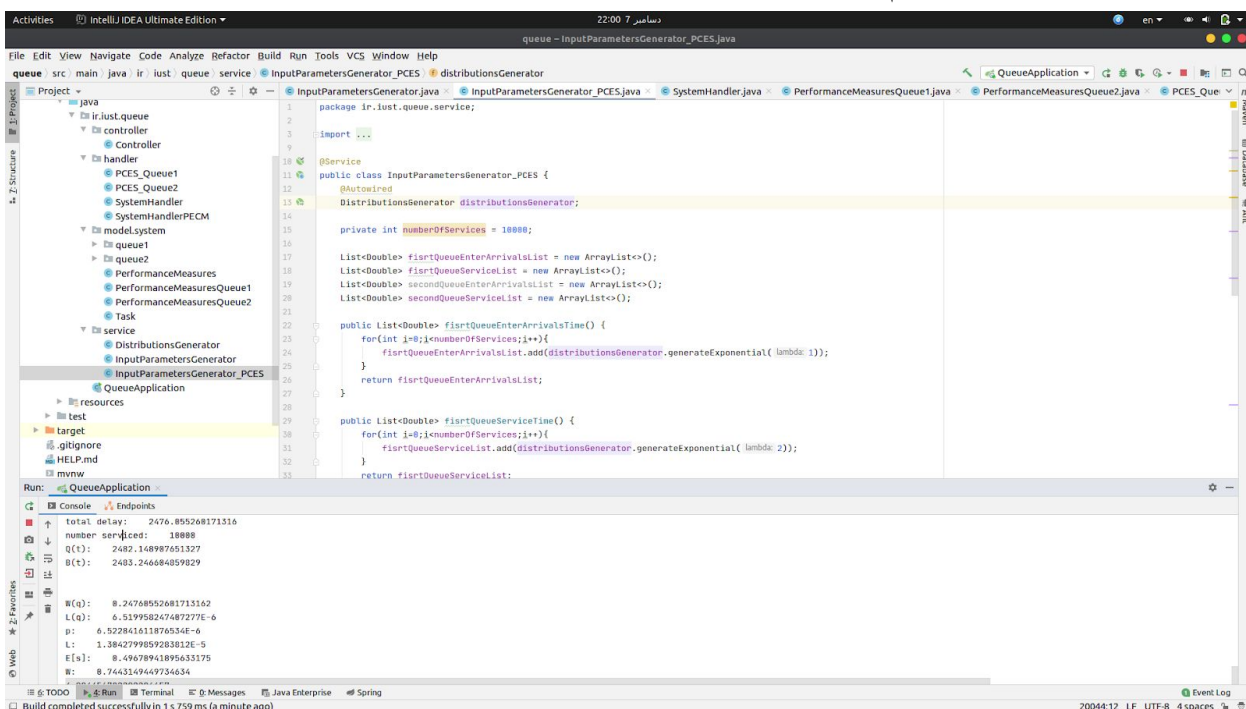
در انتها نیز معیارهای کارایی برای هر کدام از صف‌ها و نیز برای کل سیستم محاسبه می‌شود.

نتایج:

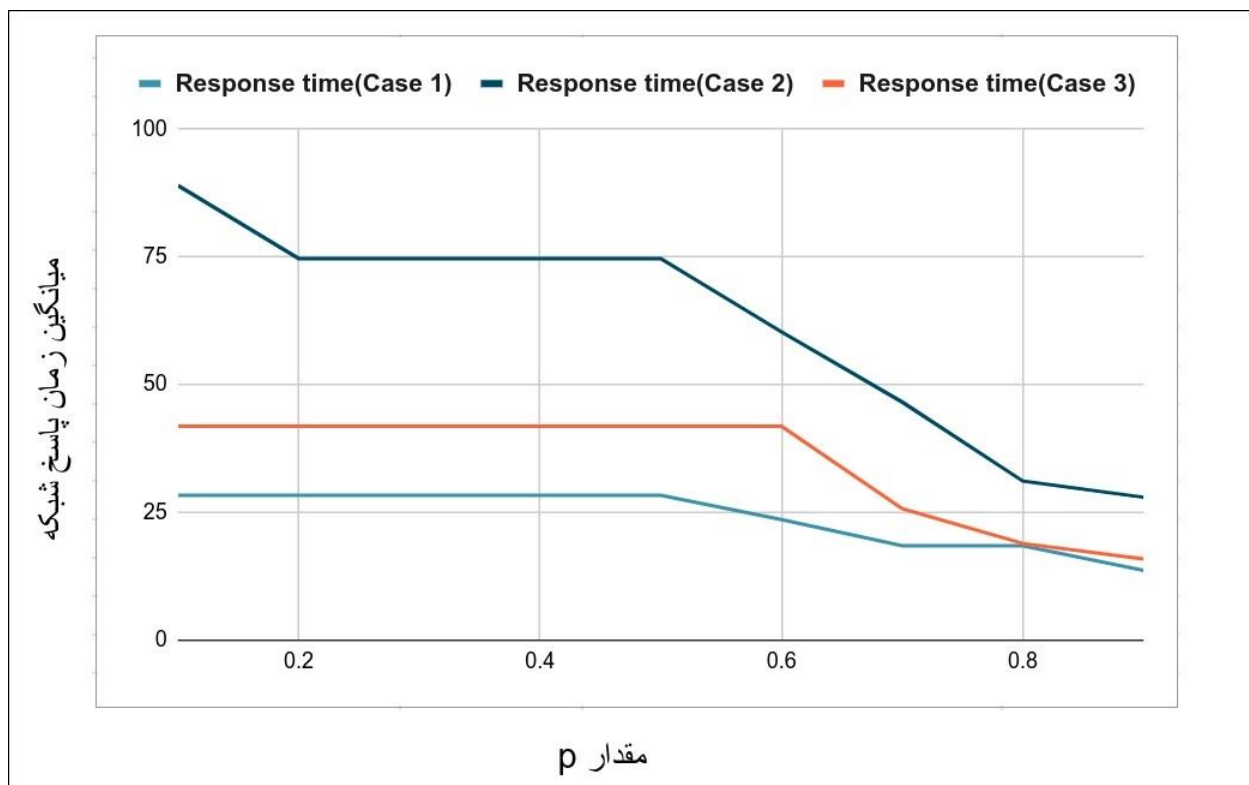
معیارهای کارایی برای صف اول در تصویر زیر قابل مشاهده است:



معیارهای کارایی برای صف دوم در تصویر زیر قابل مشاهده است:



و همچنین نمودار تاثیر پارامتر p بر زمان پاسخ شبکه نیز به شرح زیر است:



نحوه اجرای کد:

این پروژه با زبان `java` و با استفاده از فریم‌ورک `spring boot` نوشته شده است. برای اجرای پروژه می‌توانید `QueueApplication.java` را اجرا کنید و برای تست صحت عملکرد پروژه که همانطور که پیش‌تر ذکر شد از مثال حل شده اسلایدها استفاده شده است می‌توانید در کلاس `controller` دو خط زیر از تابع `test` را از کامنت خارج کرده و سایر خط‌های این تابع را کامنت کنید:

```
//      systemHandler.initializeSystemParameters();
//      systemHandler.run();
```

پس از اجرای کد وارد `browser` خود شده و آدرس زیر را وارد کنید:

`http://localhost:8080/swagger-ui.html`

بخش `controller` آن را اجرا کرده و اطلاعات مورد نیاز را در ترمینال مشاهده کنید.

بخش امتیازی:

همانطور که در بخش اجرای کد ذکر شد رابط کاربری swagger به شما امکان اجرای کد را می‌دهد.