به نام خدا

# تمرین شماره ۴ تحلیل و طراحی سیستم‌ها


دانشکده مهندسی برق و کامپیوتر

| اعضای گروه: | شماره دانشجویی: |
| --- | --- |
| حامد میرامیرخانی | ۸۱۰۱۹۹۵۰۰ |
| سینا طبسی | ۸۱۰۱۹۹۵۵۴ |
| علی عطاءاللهی | ۸۱۰۱۹۹۴۶۱ |
| محمدرضا بهفر | ۸۱۰۱۹۷۴۷۲ |


---

## الف) نمودار های تعامل «درخواست بسته درمانی»

در فاز قبلی برای «درخواست بسته درمانی» دو قرارداد عملیاتی تعریف شد که در این فاز دو قرارداد عملیات دیگر نیز به درخواست بسته درمانی اضافه می‌شود که به انتهای این گزارش اضافه شده اند.

حال به ازای هر قرارداد عملیاتی یک نمودار تعامل رسم می‌کنیم (از sequence diagram استفاده می‌کنیم).

sequence diagram of **RequestNewPackage:**

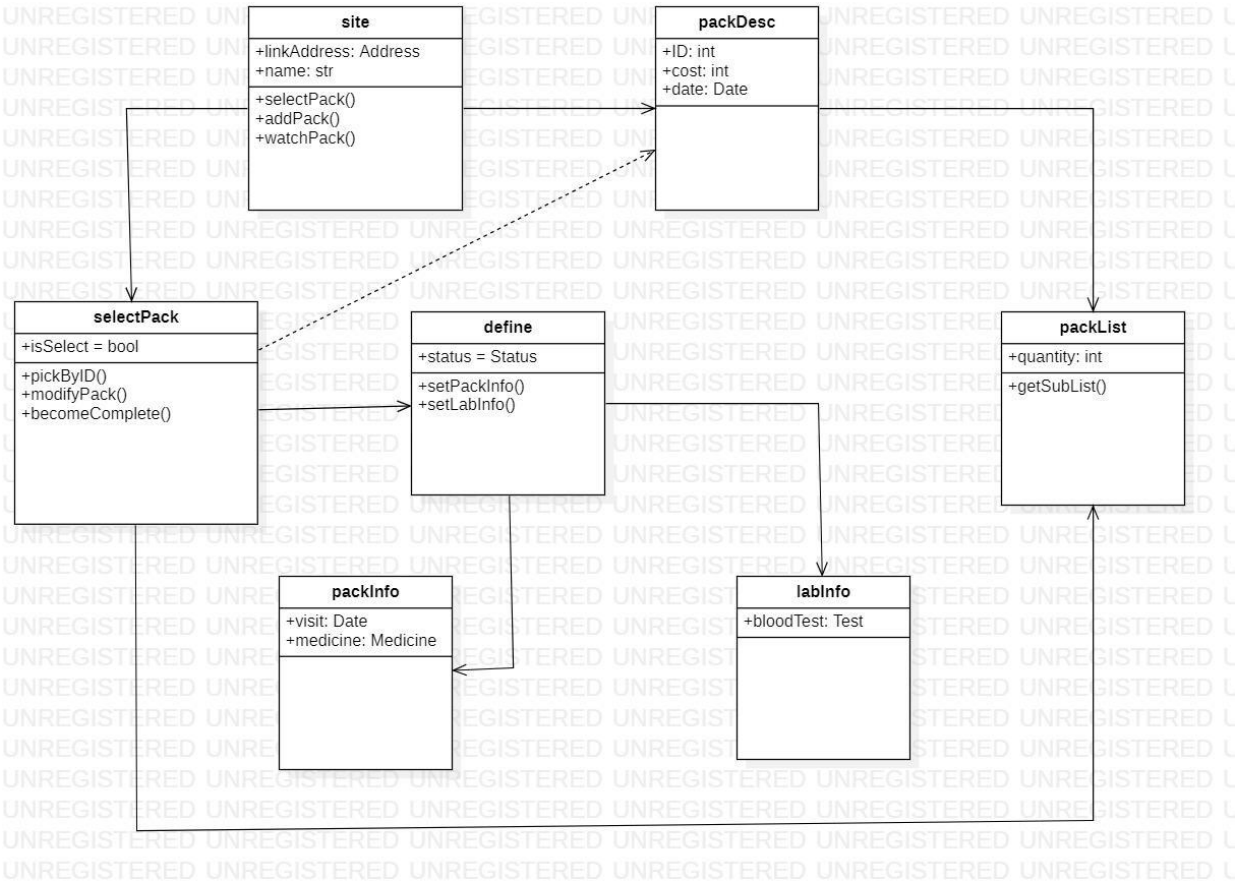
sequence diagram of **FillRequirements and RecordAssignments:**

sequence diagram of **SelectPackage:**

## ب) کلاس‌های طراحی

تعریف بسته درمانی

---



1. کلاس site:
   - ارتباط با کلاس selectPack از طریق یال Association.
   - متدها:
     - selectPack(): این متد کاربر را به کلاس selectPack هدایت می‌کند.
     - addPack(): این متد امکان اضافه کردن بسته جدید را فراهم می‌کند.
     - watchPack(): این متد امکان مشاهده بسته‌های موجود را به کاربر می‌دهد.
2. کلاس selectPack:
   - ارتباط با کلاس site و کلاس define از طریق یال‌های Association.
   - متدها:
     - pickById(): این متد امکان انتخاب بسته بر اساس شناسه را فراهم می‌کند.
     - modifyPack(): این متد امکان تغییر اطلاعات بسته را فراهم می‌کند.
     - becomeComplete(): این متد با تکمیل اطلاعات بسته، بسته را کامل می‌کند.
3. کلاس define:
   - ارتباط با کلاس selectPack و کلاس packDesc از طریق یال‌های Association.
   - متدها:
     - setPackInfo(): این متد اطلاعات بسته را تعیین می‌کند.
     - setLabInfo(): این متد اطلاعات آزمایشگاهی را تعیین می‌کند.
4. کلاس packList:
   - ارتباط با کلاس selectPack و کلاس packInfo از طریق یال‌های Association.
   - متدها:
     - getSubList(): این متد یک زیر لیست از بسته‌ها را باز می‌گرداند.

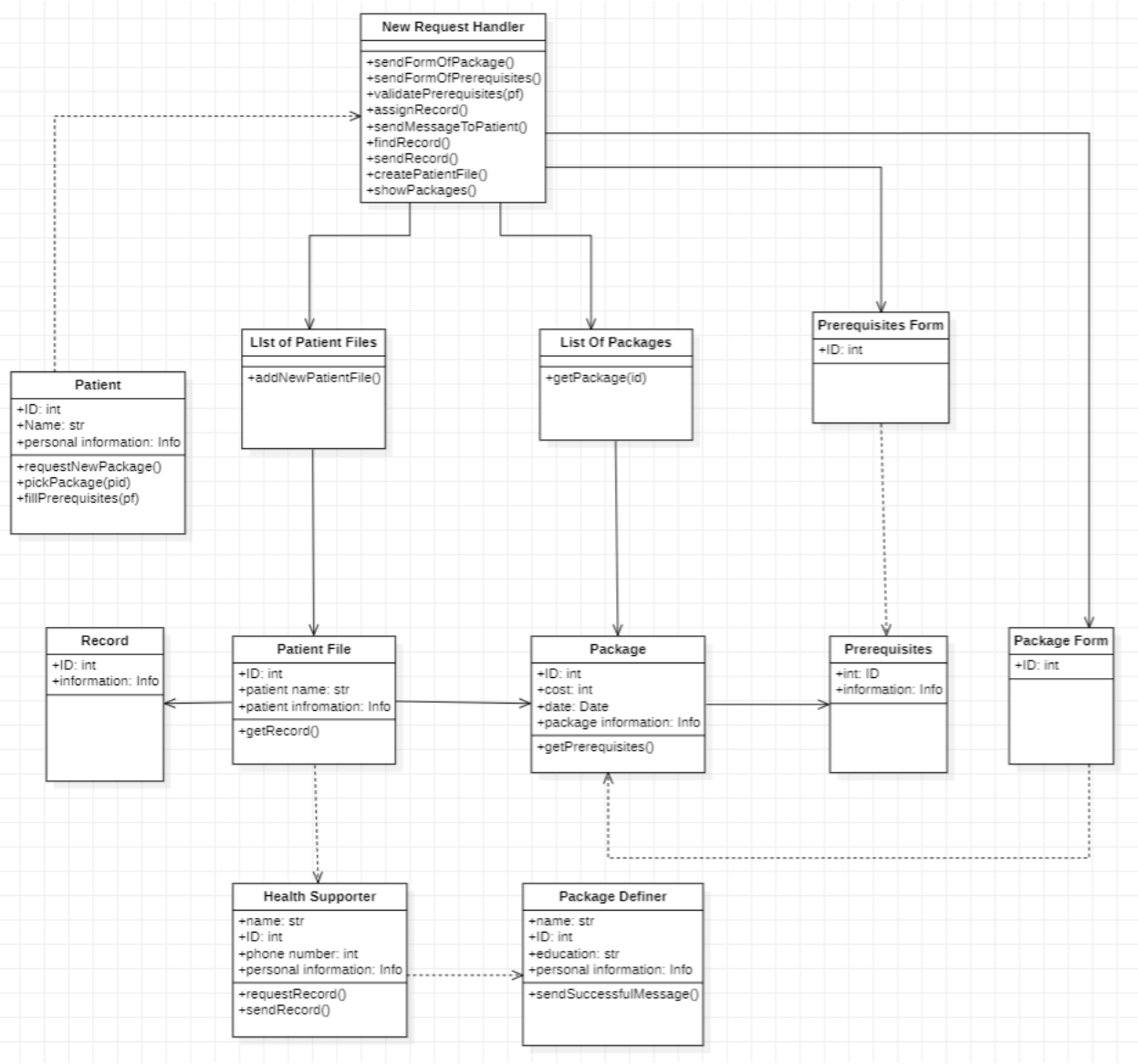
در مورد اصول طراحی GRASP در این مورد High Cohesion را می‌توان با ایجاد کلاس‌ها با مسئولیت‌های محدود و متمرکز بیان کرد. به عنوان مثال، کلاس "define" فقط برای تعیین اطلاعات بسته و اطلاعات آزمایشگاهی ساخته شده است.

Low Coupling هم با محدود کردن تعداد ارتباطات مستقیم بین کلاس‌ها ایجاد می‌شود. این اصل کمک می‌کند که تغییرات در یک کلاس کمترین تاثیر را بر دیگر کلاس‌ها داشته باشد.

از اصول دیگر GRASP می‌توان به "Information Expert" اشاره کرد که به معنای تخصیص مسئولیت به کلاسی است که بیشترین اطلاعات یا دانش را در مورد آن دارد. به عنوان مثال، در این طراحی، متد "setPackInfo()" در کلاس "define" قرار داده شده است، زیرا این کلاس دانش لازم برای تعیین اطلاعات بسته را دارد.

از سوی دیگر، اصل "Creator" نیز در این طراحی رعایت شده است. در این اصل، یک کلاس به عنوان سازنده یک نمونه از کلاس دیگر در نظر گرفته می‌شود. به عنوان مثال، کلاس "site" ایجاد کننده نمونه‌های کلاس "selectPack" است.

## درخواست بسته درمانی




1. کلاس New Request Handler: این کلاس به عنوان رابط برای مدیریت درخواست های جدید کاربران عمل می کند. توابعی که در این کلاس می باشند عبارتند از:

- sendFormOfPackage(): این تابع فرم بسته را به کاربر ارسال می کند.
- sendFormOfPrerequisites(): این تابع فرم مربوط به پیش نیازها را به کاربر ارسال می کند.
- validatePrerequisites(pf): این تابع پیش نیازهای ارسالی توسط کاربر را اعتبار سنجی می کند.
- assignRecord(): این تابع یک رکورد را به یک بیمار اختصاص می دهد.
- sendMessageToPatient(): این تابع یک پیام را به بیمار می فرستد.
- findRecord(): این تابع یک رکورد را جستجو می کند.
- sendRecord(): این تابع یک رکورد را می فرستد.
- createPatientFile(): این تابع فایل بیمار را می سازد.
- showPackages(): این تابع بسته های موجود را به بیمار نمایش می دهد.

این کلاس با کلاس های Patient و List Of Packages از طریق ارتباط dependency و با کلاس های Patient File و Health Supporter از طریق ارتباط association در ارتباط است.

2. کلاس List Of Patient Files: این کلاس فهرست فایل های بیمار را نگه داری می کند. تابع این کلاس عبارت است از:
   - addNewPatientFile(): این تابع یک فایل بیمار جدید را به لیست اضافه می کند.

این کلاس با کلاس Patient File از طریق ارتباط association و با کلاس New Request Handler از طریق ارتباط dependency در ارتباط است.

3. کلاس List Of Packages: این کلاس فهرست بسته ها را نگه داری می کند. تابع این کلاس عبارت است از:
   - getPackages(id): این تابع بسته های مربوط به یک شناسه خاص را از لیست برمی گرداند.

این کلاس با کلاس Package از طریق ارتباط association و با کلاس New Request Handler از طریق ارتباط dependency در ارتباط است.

4. کلاس Package: این کلاس یک بسته را نمایش می دهد. تابع این کلاس عبارت است از:
   - getPrerequisites(): این تابع پیش نیازهای یک بسته را برمی گرداند.

این کلاس با کلاس List Of Packages از طریق ارتباط association در ارتباط است.

5. کلاس Patient File: این کلاس یک فایل بیمار را نمایش می دهد. تابع این کلاس عبارت است از:
   - getRecord(): این تابع رکوردی را از فایل بیمار برمی گرداند.

این کلاس با کلاس های List Of Patient Files و New Request Handler از طریق ارتباط association در ارتباط است.

6. کلاس Patient: این کلاس یک بیمار را نمایش می دهد. توابع این کلاس عبارتند از:
   - requestNewPackage(): این تابع یک درخواست بسته جدید را ایجاد می کند.
   - pickPackage(pid): این تابع یک بسته خاص را انتخاب می کند.

- fillPrerequisites(pf): این تابع پیش نیاز های یک بسته را پر می کند.

این کلاس با کلاس New Request Handler از طریق ارتباط dependency در ارتباط است.

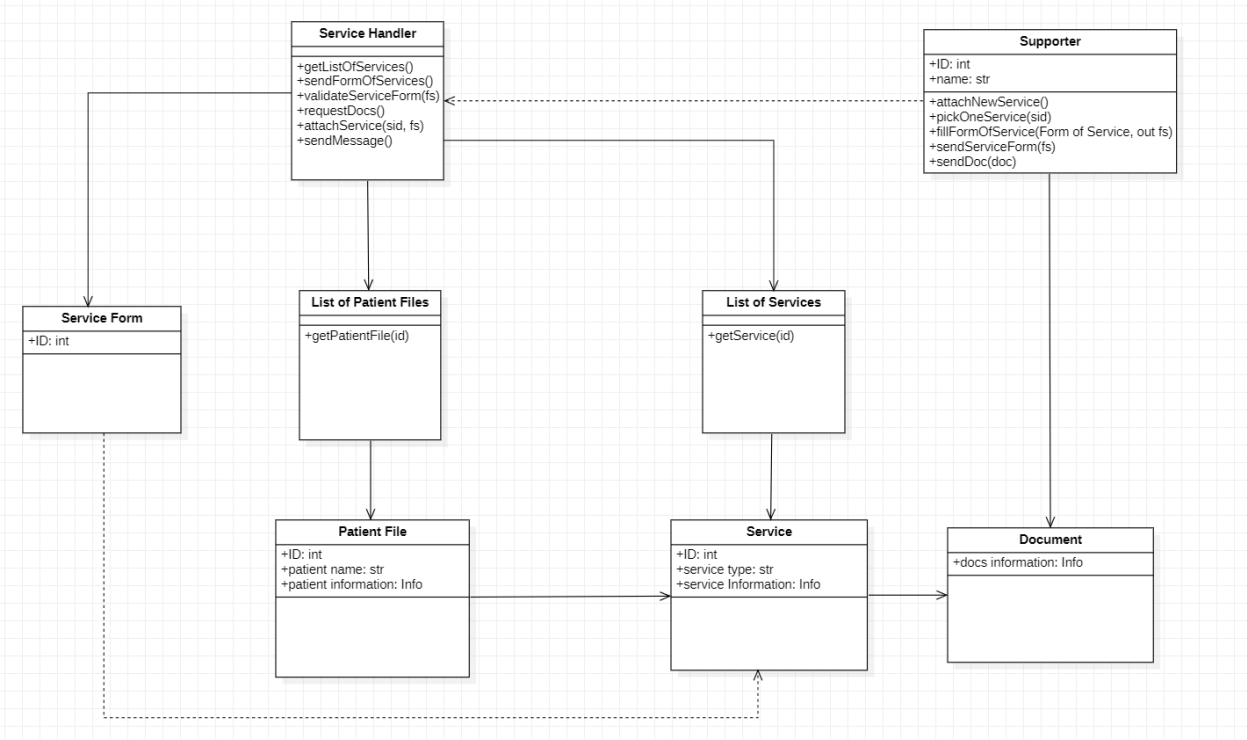
7. کلاس Package Definer: این کلاس بسته را تعریف می کند. تابع این کلاس عبارت است از:
   - sendSuccessfulMessage(): این تابع یک پیام موفقیت آمیز ارسال می کند.

این کلاس با کلاس Patient از طریق ارتباط dependency در ارتباط است.

8. کلاس Health Supporter: این کلاس یک حامی سلامت را نمایش می دهد. توابع این کلاس عبارتند از:
   - requestRecord(): این تابع یک درخواست رکورد از سیستم می کند.
   - sendRecord(): این تابع یک رکورد را به سیستم ارسال می کند.

این کلاس با کلاس های Patient File و New Request Handler از طریق ارتباط association در ارتباط است.

## الصاق خدمت جدید به پرونده بیمار



کلاس ها:

ServiceHandler: این کلاس وظیفه ارتباط با سرویس ها را دارد و مسئولیت های مربوط به خدمات به این کلاس محول شده است.

توابع:

getListOfServices(): این تابع لیست کلیه خدمات موجود را دریافت می کند.

sendFormOfServices(): این تابع فرم مربوط به خدمات را ارسال می کند.

validateServiceForm(fs): این تابع فرم خدمات را اعتبار سنجی می کند.

requestDocs(): این تابع مستندات مربوط به خدمات را درخواست می کند.

attachServices(sid, fs): این تابع خدمات را به فایل بیمار الصاق می کند.

sendMessage(): این تابع پیامی را ارسال می کند.

Supporter: این کلاس نماینده یک پشتیبان است که مسئولیت الصاق خدمت جدید به پرونده بیمار را دارد.

توابع:

attachNewService(): این تابع یک خدمت جدید را الصاق می کند.

pickOneService(sid): این تابع یک خدمت را از بین خدمات موجود انتخاب می کند.

fillFormOfSerivce(Form of Service, our fs): این تابع فرم مربوط به خدمت را پر می کند.

sendServiceForm(fs): این تابع فرم خدمت را ارسال می کند.

sendDoc(doc): این تابع مستندات را ارسال می کند.

ListOfServices: این کلاس لیست خدمات را نگهداری می کند.

توابع:

getService(id): این تابع خدمتی را با استفاده از شناسه خدمت دریافت می کند.

ListOfPatientFiles: این کلاس لیست فایل‌های بیماران را نگهداری می کند.

توابع:

getPatientFile(id): این تابع فایل یک بیمار را با استفاده از شناسه بیمار دریافت می کند.

روابط بین کلاس ها:

بین کلاس ها روابطی به نظر می رسد که با توجه به عملکرد هر کلاس و توابعی که در آن ها تعریف شده، تعریف می شوند. در زیر روابط محتمل را بیان کرده‌ام:

ServiceHandler به ListOfServices: ServiceHandler نیاز به دسترسی به ListOfServices دارد تا بتواند لیست خدمات موجود را دریافت کند. این رابطه یک ارتباط تک به چند است.
ServiceHandler به ListOfPatientFiles: ServiceHandler نیاز به دسترسی به ListOfPatientFiles دارد تا بتواند خدمات را به فایل بیمار الصاق کند. این نیز یک رابطه تک به چند است.
Supporter به ServiceHandler: Supporter نیاز به دسترسی به ServiceHandler دارد تا بتواند خدمت را انتخاب کند، فرم خدمت را پر کند و آن را ارسال کند. این رابطه یک رابطه یک به یک است.

اتخاذ تصمیمات:

روابط و توابع کلاس‌ها بر اساس وظایفی که هر کلاس باید انجام دهد، تعیین شده اند. برای نمونه، اگر یک کلاس نیاز به دسترسی به داده‌های یک کلاس دیگر دارد، یک رابطه بین آن‌ها تعریف می‌شود. توابع هر کلاس نیز براساس کارکردهایی که آن کلاس باید انجام دهد، انتخاب شده اند. برای نمونه، Supporter نیاز به تابع attachNewService() دارد تا بتواند یک خدمت جدید را الصاق کند.

Supporter:

در این کلاس، تمام توابع مرتبط با رفتار کارشناس سلامت (یا همان پشتیبان) هستند. این توابع شامل انتخاب خدمت، تکمیل فرم خدمت و ارسال آن و ارسال مدارک مربوطه هستند. این تصمیم بر اساس اصل Single Responsibility (یک وظیفه‌ای) در طراحی سیستم‌های نرم‌افزاری گرفته شده است که بیان می‌کند هر کلاس باید تنها یک وظیفه را بر عهده داشته باشد.

ServiceHandler:

این کلاس وظیفه مدیریت خدمات را بر عهده دارد. توابع آن شامل دریافت لیست خدمات، ارسال فرم خدمات، اعتبارسنجی فرم خدمات، درخواست مدارک و الصاق خدمات به پرونده بیمار هستند. تصمیم برای این کلاس نیز بر اساس اصل Single Responsibility گرفته شده است.

ListOfServices و ListOfPatientFiles:

این دو کلاس به عنوان مخازن داده‌ای عمل می‌کنند که اطلاعات خدمات و فایل‌های بیمار را نگهداری می‌کنند. این تصمیم بر اساس اصل Separation of Concerns (جداسازی نگرانی‌ها) در طراحی نرم‌افزار گرفته شده است که بیان می‌کند بخش‌های مختلف سیستم باید جدا از هم عمل کنند.

# تغییرات اعمال شده

| نام قرارداد: **RequestNewPackage** | |
|---|---|
| RequestNewPackage() | نام عملیات |
| درخواست بسته درمانی :Use Cases | مورد کاربرد مرجع |
| - بسته از قبل برای درخواست بیمار شکل گرفته باشد.<br>- بسته ها در دیتابیس سایت ذخیره شده باشد .<br>- تاییدیه بسته درمانی توسط گروهی از پزشکان صادر شده باشد. | پیش شرط |
| - لاگ مربوط به فعالیت های User مورد نظر بروزرسانی شد.<br>- صف درخواست ها در system بروزرسانی شد.<br>- درخواستی جدید درsystem به وجود آورده شد. | پس شرط |

- همچنین به مورد مرجع کاربرد **FillRequirements**در فاز قبلی درخواست بسته درمانی هم اضافه می‌شود.