دانشکده ی مهندسی برق و کامپیوتر دانشگاه یزد

پایان نامه جهت اخذ درجه کارشناسی در رشته مهندسی برق _ مخابرات

عنوان

طراحی اپلیکیشن اندروید برای تشخیص سیگنال ناسالم قلب

استاد راهنما

دکتر جمشید ابوئی

پژوهش و نگارش

مصطفی قراخانلو

تیر ماه ۱۳۹۸

بسمه تعالی

اکنون که این پروژه به انتها رسیده است،پس از حمد و سپاس به درگاه باری تعالی لازم می بینم تا از زحمات تمامی کسانی که در به انجام رسانده این پروژه به بنده کمک نمودند تشکر نمایم.

از اساتید محترم جناب آقای دکتر جمشید ابویی و جناب آقای دکتر وحید ابوطالبی و از دوستان عزیزم به خصوص جناب آقای مهندس امیرحسین مهدی پور که کمک و همیاری شایانی به اینجانب نمودند به خاطر زحمات بی بدیل و راهنمایی های مفیدشان قدردانی می نمایم.

مصطفی قراخانلو

تابستان ۱۳۹۸

فهرست عناوین

فصل اول : مقدمه

فصل دوم: نرم افزار

فصل سوم: پیشنهادات و منابع

فـصل اول :

مـقـــد مـــه

۱-۱) مقدمه :

بیماری قلبی یکی از رایج ترین علل مرگ و میر در جوامع امروزی به شمار می رود.تغییر ساختار زندگی مردم، عدم تحرک کافی و تغذیه نا مناسب همه و همه از عوامل بروز این گونه بیماری ها به شمار می روند.

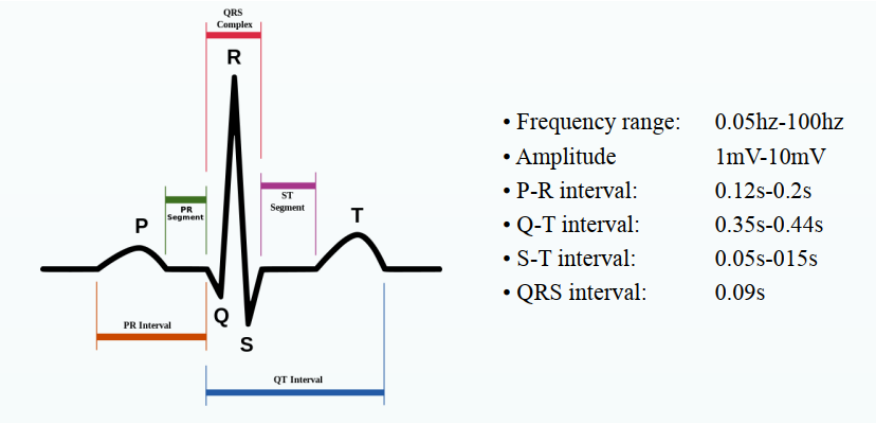
گونه ای از بیماری های قلبی به این علت به وجود می آیند که ضربان قلب ریتم طبیعی خود را از دست داده است که از آن به عنوان آریتمی قلبی یاد می شود.

هدف این است که به وسیله یک اپلیکیشن اندروید، ناهنجاری های قلبی سیگنال قلب تشخیص داده شود. بر این اساس اپلیکیشنی بر بستر اندروید به منظور تشخیص سه نوع مهم از ناهنجاری های قلبی که تاکی کاردی ، برادی کاردی و آریتمی نام دارند طراحی می شود.

لازم است قبل از توضیح ناهنجاری های قلبی ابتدا سیگنال قلب به طور مختصر معرفی شود تا با ویژگی های این سیگنال آشنا شویم.

۱-۲) معرفی سیگنال های قلبی :

الکتروکاردیوگرام (Electrocardiogram) یا نوار قلب که معمولا با مخفف ECG مشخص می شود، به نمودار ثبت شده تغییرات سیگنال الکتریکی ناشی از تحریک عضله قلب گفته می شود.در شکل زیر الگوی یک پالس قلبی به همراه مقادیر مربوط به یک قلب سالم برای هر یک بازه های زمانی ، دامنه و فرکانس نشان داده شده است.تعداد ضربان یک قلب سالم بین ۶۰ تا ۱۰۰ تپش در دقیقه می باشد.



از روی این منحنی می توانیم به نحوه عملکرد قلب پی ببریم. هر منحنی شامل ۳ موج است:

۱- موج P که فعالیت الکتریکی دهلیز ها را نمایش می دهد.

۲-موج QRS که فعالیت الکتریکی بطن ها را نمایش می دهد.

۳-موج T که نمایش دهنده استراحت بطن ها می باشد

موج QRS چون دارای تغییرات سریع و دامنه ای بزرگ می باشد ، از بقیه امواج قابل تشخیص تر است. بنابراین میتوان با شمردن تعداد امواج QRS یا به عبارت بهتر R-Wave ها تعداد ضربان قلب را معلوم کرد. با داشتن فواصل بین ضربان قلب می توان تند زدن یا کند زدن غیرمعمول قلب را تشخیص داد. همچنین با بررسی فاصله وقوع دو QRS می توان وجود آریتمی را در فرد بررسی کرد. علاوه بر این میتوان بعد از تعیین QRS ، بقیه امواج ECG مانند موج P و موج Tرا نیز تعیین کرد و از آنها برای تشخیص بیماری های پیچیده تر نیز استفاده نمود.

البته نباید فراموش کرد که همه ی بیماری های قلبی از روی ECG قابل تشخیص نمی باشد و برای تشخیص بعضی از بیماری ها به آزمایشات پیچیده تری نیاز است.

در این پروژه بر بستر اندروید به شناسایی سه نوع از رایج ترین ناهنجاری های قلبی پرداخته ایم که هر سه بر مبنای فاصله زمانی بین R-Peak ها قابل تشخیص هستند. در قسمت بعدی این ناهنجاری ها را معرفی نموده و ضمن توضیحاتی مختصر در مورد آنها ، مشخصه های آنها را نیز بیان می کنیم.

۱-۳) معرفی ناهنجاری های قلبی مورد بررسی قرار گرفته در پروژه :

۱-۳-۱) تاکی کاردی : تاکی کاردی یا تندتپشی هنگامی رخ می دهد که تعداد ضربان قلب از ۱۰۰ ضربان در دقیقه بیشتر شود.البته تندتپشی همیشه یک بیماری نیست بلکه گاه به صورت فیزیولوژیک در پاسخ نیاز به بدن به اکسیژن در مواقعی مانند ورزش ایجاد می شود.

۱-۳-۲) برادی کاردی : برادی کاردی یا کندتپشی هنگامی رخ می دهد که تعداد ضربان قلب از ۶۰ ضربان در دقیقه کمتر شود. البته کندتپشی اگر در محدوده بین ۵۰ تا ۶۰ ضربان در دقیقه باشد معمولا بدون علامت است اما وقتی ضربان کمتر شود در فرد نشانه هایی مانند سنکوپ ، سرگیجه ، ضعف ، خستگی ، نفس های کوتاه ، درد قفسه سینه ، اختلال خواب و گیجی را ایجاد می کند.

۱-۳-۳) آریتمی : آریتمی با نام دیگر کژآهنگی به غیرطبیعی بودن ریتم قلب گفته می شود. ریتم طبیعی قلب از گره سینوسی آغاز می شود و پس از انتقال به گره دهلیزی-بطنی در بطن ها انتشار می یابد در نتیجه بر مبنای این هدایت تحریک الکتریکی ، ابتدا دهلیزها و سپس با فاصله کمی بطن ها منقبض می شوند. واضح است که آریتمی از این مبنا پیروی نمی کند.

در جدول زیر ناهنجاری های مورد بررسی و مشخصه آنها نشان داده شده است:

ناهنجاری	مشخصه
Tachycardia	فاصله بین دو R-Peak کمتر از 0.6s
Bradycardia	فاصله بین دو R-Peak بیش از 1s
Arrhythmia	فواصل زمانی وقوع R-Peak ها نامنظم

۱-۴)داده های استفاده شده در پروژه:

در این پروژه از نمونه های ECG موجود در MIT- BIH Arrhythmia Database واقع در سایت Physionet استفاده شده است. این سیگنال ها دارای فرمت dat. بوده و نمونه برداری آن در هر کانال (برای نمونه برداری سیگنال از دو کانال استفاده شده است) با فرکانس ۳۶۰ هرتز و رزولوشن ۱۱ بیتی و محدوده ولتاژی ۱۰ میلی ولت انجام شده است. هر یک سیگنال های ECG یاد شده حدود ۶۵۰۰۰۰ نمونه دارد و اندازه زمانی آنها حدودا ۳۰ دقیقه است.

۱-۵) روش شناسایی R-Peak ها :

اگرچه ساده ترین راه برای شناسایی R-Peak ها به دست آوردن ماکزیمم دامنه ی سیگنال است اما این روش برای پیاده سازی نرم افزار کارآمد نیست زیرا دامنه این سیگنال ها متغیر است و همچنین مقدار کمی دارد لذا تعیین مقدار آستانه بیشینه ولتاژ برای تمام شرایط ممکن نیست.پس لازم است از الگوریتمی استفاده شود که توانایی پیاده سازی آن نیز بر بستر اندروید نیز امکان پذیر باشد.در این پروژه ما از الگوریتم تشخیص QRS بر پایه مشتق که توسط Balda ارائه شده است استفاده کرده ایم.

۱-۵-۱) الگوریتم Balda :

در این الگوریتم ابتدا مشتقی سه نقطه ای و هموار شده بر روی داده $x(n)$ صورت می گیرد که با معادله ی زیر قابل توصیف می باشد و آن را با $y_0(n)$ نمایش می دهیم:

$$y_0(n) = |x(n) - x(n-2)|.$$

سپس دومین اشتقاق به وسیله معادله زیر بر روی داده انجام می پذیرد و آن را با $y_1(n)$ نمایش می دهیم:

$$y_1(n) = |x(n) - 2x(n-2) + x(n-4)|.$$

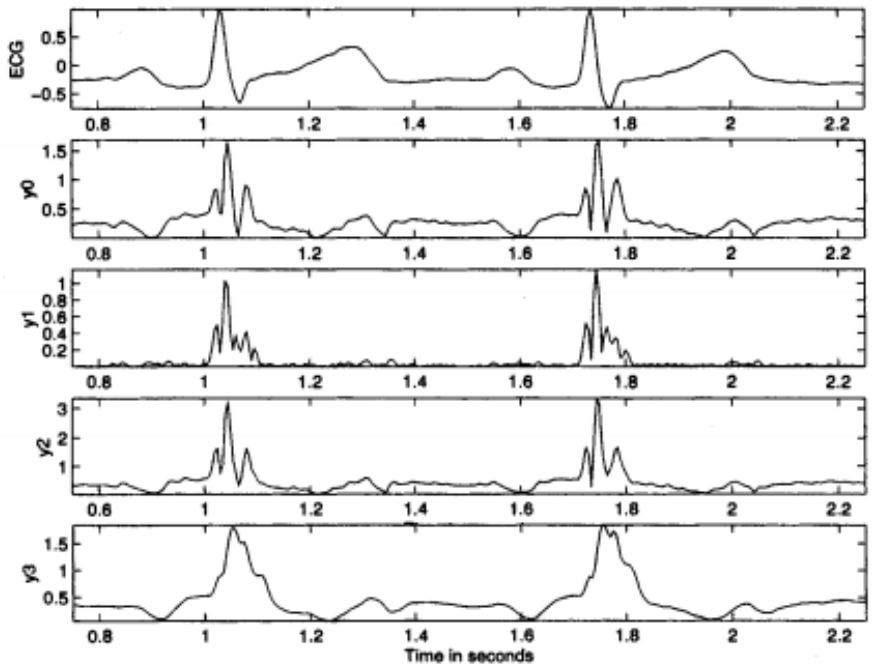
نتایج دو معادله ی فوق وزن داده شده و با هم دیگر ترکیب می شوند تا معادله ی زیر بدست بیاید:

$$y_2(n) = 1.3y_0(n) + 1.1y_1(n).$$

و در نهایت از یک فیلتر متوسط شناور (Moving Average) ۸ نقطه ای استفاده خواهیم کرد. فیلتر مذکور به شرطی که ورودی $x(n)$ و خروجی $y(n)$ باشد با معادله زیر نمایش داده می شود:

$$y(n) = \frac{1}{8}\sum_{k=0}^{7} x(n-k).$$

نمونه ای از عملکرد این الگوریتم روی یک سیگنال مفروض ECG در شکل زیر نشان داده شده است.



قله های سیگنال y3(n) حاصل شده بر اساس این الگوریتم همان R-Peak های مورد نظر ما می باشند که با استفاده از این الگوریتم دقت آستانه گذاری ما را بهتر خواهند کرد.

در نهایت با محاسبه فاصله میان قله ها و با توجه به مقادیر موجود در جدول ناهنجاری های مورد بررسی ، می توان نوع ناهنجاری را تشخیص داد.

فصل دوم :

نرم افزار

۲-۱) مقدماتی درباره ی سیستم عامل اندروید :

اندروید یک سیستم‌عامل «متن‌باز (open source) «است. عبارت «متن‌باز» بدین معناست که کد بنیادی این سیستم‌عامل، رایگان محسوب می‌شود و هرکسی می‌تواند بخشی از کدها را با توجه به نیازهای خود تغییر دهد. به دلیل متن‌باز بودن و مشخصه‌های انعطاف‌پذیر این سیستم‌عامل، حضور اندروید محدود به گوشی‌ها نمانده است و گجت های مختلفی از جمله کنسول های بازی ، یخچال های هوشمند و دیگر موارد از اندروید بهره می برند.

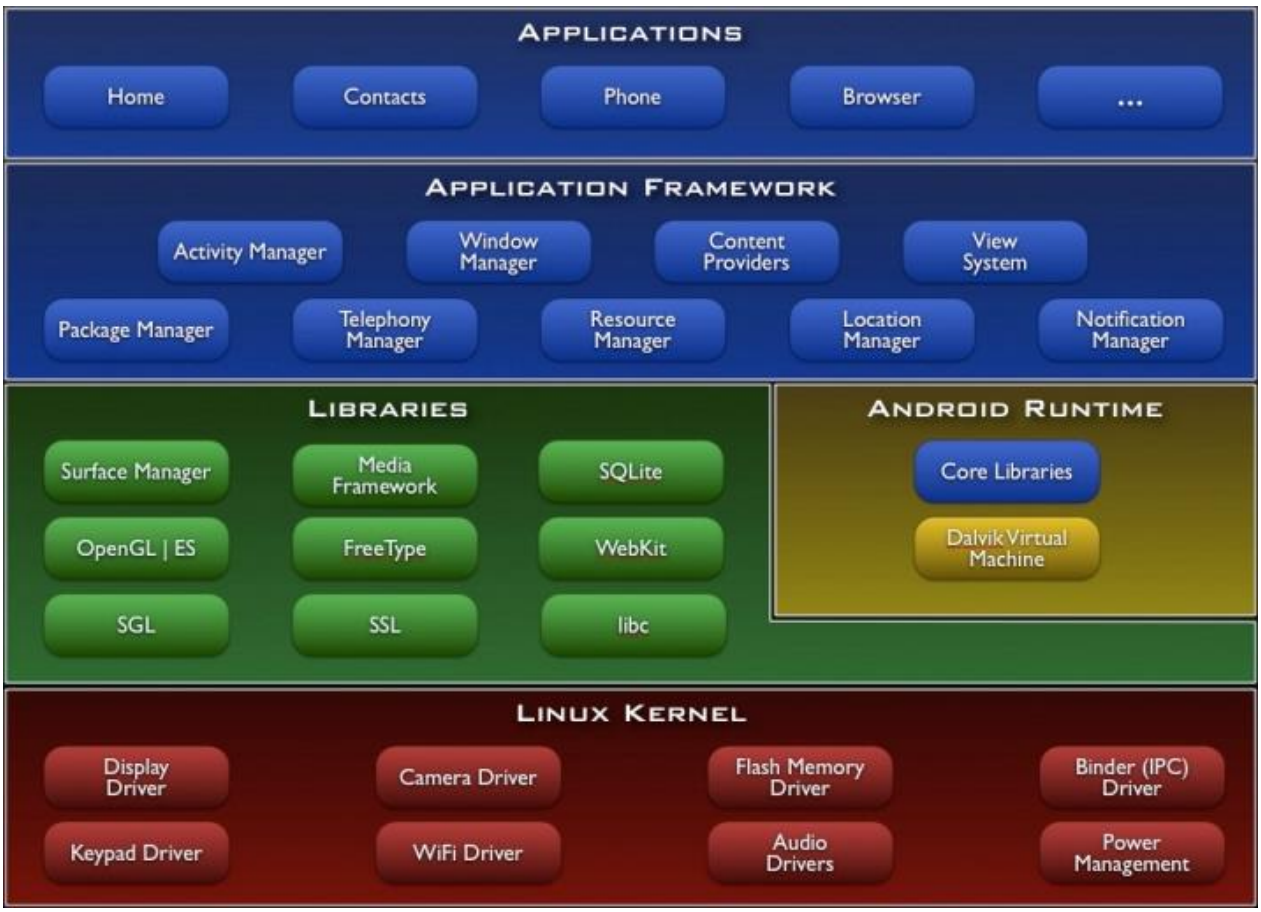
اکثر برندهای تولید کننده گوشی های هوشمند از سیستم عامل اندروید بهره می گیرند. هسته اصلی سیستم عامل تکامل یافته اندروید از سیستم عامل لینوکس گرفته شده است. در واقع می توان اندروید را برگرفته از لینوکس دانست گرچه تفاوت های بسیاری وجود دارد اما مهم این است که ساختار اندروید بر اساس لینوکس است. جالب است بدانید که درصد گوشی های استفاده کننده از اندروید بسیار بالاست و در رتبه نخست قرار دارد. معمولا روال کار به این شکل است که شرکت های تولید کننده گوشی های هوشمند که قصد استفاده از سیستم عامل اندروید در گوشی های مورد تولید خود دارند مبلغی را به گوگل جهت برخورداری از استفاده قانونی از این سیستم عامل پرداخت می کنند. زبان برنامه نویسی اندروید بر پایه Java , C , C++ می باشد.

اندروید از سال ۲۰۰۸ بصورت رسمی منتشر و تا به حال نسخه های متعددی از آنها در دسترس قرار گرفته است. در زیر به نام، معنی و تاریخ نسخه های اندروید اشاره شده است:

- نسخه Alpha به معنی آلفا – منتشر شده در تاریخ ۲۳ دسامبر ۲۰۰۸
- نسخه Beta به معنی بتا – منتشر شده در تاریخ ۹ فوریه ۲۰۰۹
- نسخه Cupcake به معنی کیک فنجانی – منتشر شده در تاریخ ۳۰ آپریل ۲۰۰۹
- نسخه Donut به معنی کیک دونات – منتشر شده در تاریخ ۱۵ سپتامبر ۲۰۰۹
- نسخه Eclair به معنی نان خامه ای – منتشر شده در تاریخ ۲۶ اکتبر ۲۰۰۹
- نسخه Froyo به معنی ماست یخ زده – منتشر شده در تاریخ ۲۰ می ۲۰۱۰
- نسخه Gingerbread به معنی نان زنجبیلی – منتشر شده در تاریخ ۶ دسامبر ۲۰۱۰
- نسخه Honeycomb به معنی کندوی عسل – منتشر شده در تاریخ ۲۲ فوریه ۲۰۱۱
- نسخه Ice Cream Sandwich به معنی بستنی حصیری – منتشر شده در تاریخ ۱۹ اکتبر ۲۰۱۱
- نسخه Jelly Bean به معنی آب نبات ژله ای – منتشر شده در تاریخ ۹ ژوئن ۲۰۱۲
- نسخه KitKat به معنی کیت کت – منتشر شده در تاریخ ۳۱ اکتبر ۲۰۱۳
- نسخه Lollipop به معنی آب نبات چوبی – منتشر شده در تاریخ ۱۲ نوامبر ۲۰۱۴
- نسخه Marshmallow به معنی مارشمالو – منتشر شده در تاریخ ۵ اکتبر ۲۰۱۵
- نسخه Nougat به معنی نوقا – منتشر شده در تاریخ ۲۲ آگوست ۲۰۱۶
- نسخه Oreo به معنی اوریو – منتشر شده در تاریخ ۲۱ آگوست ۲۰۱۷

- نسخه Pie به معنی پای – منتشر شده در تاریخ ۸ مارس ۲۰۱۸

گفته شد که زبان برنامه نویسی اندروید بر پایه JAVA , C++ , C می باشد و در این پروژه نیز از زبان برنامه نویسی جاوا در محیط توسعه ی Android Studio استفاده شده است.معماری اندروید در تصویر زیر قابل مشاهده است.



۲-۲) زبان برنامه نویسی جاوا :

زبان برنامه نویسی جاوا توسط جیمز گاسلینگ و سایر همکارانش در شرکت Sun Microsystems توسعه داده شد و در سال ۱۹۹۵ به دنیا معرفی شد. در ابتدا، نام این زبان برنامه‌نویسی بلوط بود اما در انتهای این پروژه، این نام به جاوا تغییر پیدا کرد .

تاریخچه انتشار این زبان از سال آغازین انتشار (۱۹۹۵) تاکنون به صورت زیر است:

- نسخه اولیه جاوا، ۱,۰ و ۱,۱ در سال ۱۹۹۶ برای سیستم عامل‌های ویندوز، سولاریس، مک و ویندوز منتشر شد.

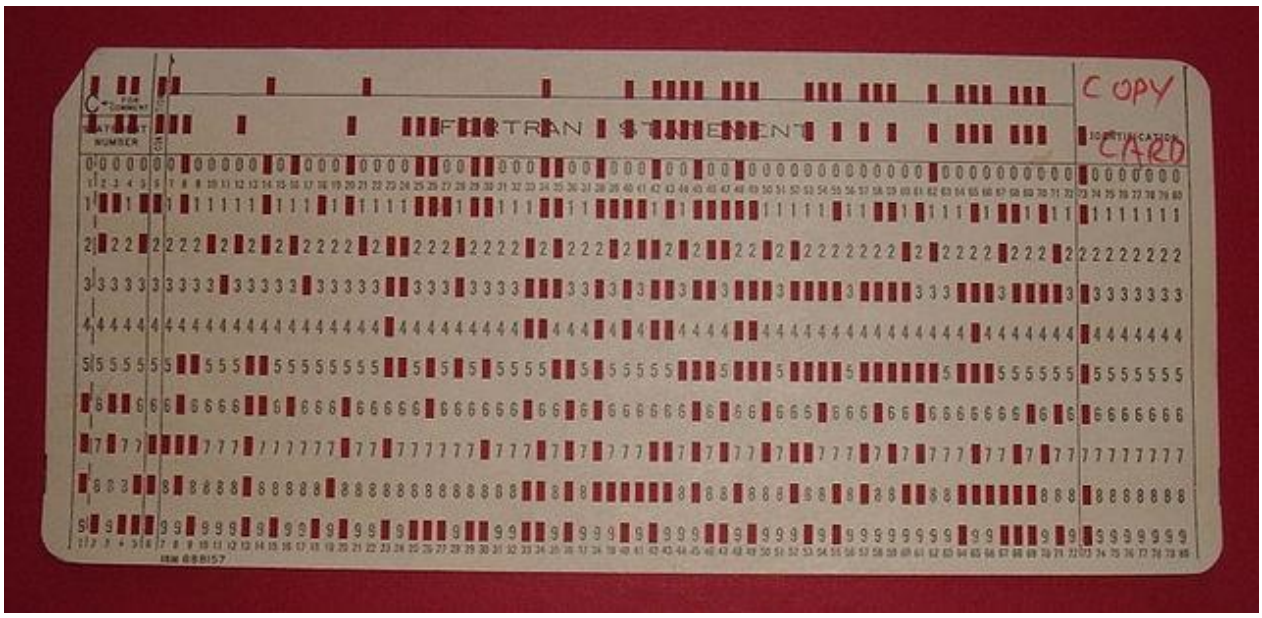
- نسخه ۱,۲ جاوا، در سال ۱۹۹۸ منتشر شد. این نسخه، نسبت به نسخه پیشین تغییرات زیادی داشت و به همین دلیل توسعه دهندگان آن را جاوا ۲ (J2SE) نامیدند.
- نسخه ۱,۳ به Kerstel معروف است و در سال ۲۰۰۰ منتشر شد.
- نسخه ۱,۴ با نام مرلین، در سال ۲۰۰۲ منتشر شد
- نسخه ۱,۵ از پلت فرم استاندارد جاوا، که با نام Tiger یا ببر شناخته شده است، در سال ۲۰۰۴ منتشر شد.
- نسخه ۱,۶ از پلت فرم استاندارد جاوا با نام Mustang در سال ۲۰۰۶ منتشر شد.
- نسخه ۱,۷ از پلت فرم استاندارد این زبان در سال ۲۰۱۱ و با نام دلفین منتشر شد.
- نسخه ۱,۸ از پلت‌فرم استاندارد جاوا در سال ۲۰۱۵ منتشر شده است.
- نسخه ۱,۹، جدیدترین نسخه این زبان، در سال ۲۰۱۸ منتشر شده است.

زبان برنامه نویسی جاوا به شهرت زیادی دست پیدا کرده و یکی از علل اصلی آن شی‌گرا بودن این زبان است. استفاده از مفهوم شی‌گرایی، توسعه نرم افزار را ساده‌تر می‌کند. برای آشنایی بیشتر با شی‌گرایی، باید با مفاهیمی همچون انتزاع، کپسوله کردن، وراثت و چندریختی آشنا بود که بعد از معرفی ویژگی های دیگر از جاوا ، در بخش بعدی درباره برنامه نویسی شی گرا و مفاهیم آن توضیح خواهیم داد.

- **قابل حمل بودن :** برنامه‌های جاوا، قابلیت حمل در شبکه را دارند. یعنی می‌توان آن را یکبار نوشت و سپس روی ماشین‌های مختلف اجرا کرد. برنامه جاوا پس از کامپایل به دستوراتی تبدیل می‌شوند که برای ماشین مجازی جاوا قابل فهم است. این دستورات بایت کد نام دارد. ماشین مجازی جاوا هم بایت کد را به کد ماشین که برای سخت افزار کامپیوتر قابل فهم است، تبدیل می‌کند. بنابراین بایت کدها را می‌توان در هر جای شبکه، روی هر کلاینت یا سروری که یک ماشین مجازی جاوا دارد، اجرا کرد.
- **مقاوم بودن** : کد های جاوا مقاوم هستند. مقاوم بودن به این معنی است که کمتر اتفاق می‌افتد که برنامه درهم شکسته شود یا اصطلاحا crash کند. برای محقق کردن این هدف، ماشین مجازی جاوا بررسی‌های خاصی را روی نوع هر شی انجام می‌دهد تا از یکپارچگی آن اطمینان حاصل کند، اشیا جاوا تنها می‌توانند به اشیا واقعی ارجاع دهند و نه به هر جای دلخواه از حافظه. مکانیزم‌های قوی جاوا برای تخصیص و آزادسازی خودکار حافظه و مدیریت خطا نیز بر مقاوم بودن برنامه‌ها اضافه می‌کند.
- **یادگیری سریع :** با توجه به اینکه نحو این زبان شبیه C++ است، یادگیری آن نسبتا آسان است، به ویژه برای افرادی که قبلا با C و C++ کار کردند.

۲-۳)برنامه نویسی شی گرا :

برای این که بدانیم برنامه‌نویسی شیءگرا چیست، ابتدا باید نقیض آن را بشناسیم. زبان‌های برنامه‌نویسی اولیه به صورت رویه‌ای بودند. دلیل این نامگذاری آن بود که در این زبان‌ها، برنامه‌نویس باید مجموعه خاصی از رویه‌ها را تعریف می‌نمود که رایانه آن‌ها را به ترتیب اجرا می‌کرد.



در زمان‌های نخست برنامه‌نویسی، رویه‌ها بر روی کارت‌های پانچ نوشته می‌شدند. رایانه‌ها بدین ترتیب داده‌ها را گرفته، یک توالی از اقدامات را بر روی داده‌ها انجام داده و سپس داده‌های جدید را در خروجی ارائه می‌کردند.

زبان‌های رویه‌ای تا مدت‌ها به خوبی کار می‌کردند و برخی از آن‌ها نیز همچنان مورد استفاده قرار می‌گیرند. اما زمانی که قرار بود برنامه‌نویس کاری را خارج از ترتیب مقدماتی مراحل انجام دهد، مدیریت زبان‌های برنامه‌نویسی دشوار می‌شد. بدین ترتیب زبان‌های برنامه‌نویسی شیءگرا وارد عرصه شدند.

نخستین زبان برنامه‌نویسی شیءگرا (که عموماً اعتقاد بر این است Simula بوده است) ایده اشیا را معرفی کرد. اشیا مجموعه‌ای از اطلاعات هستند که به عنوان واحدی منفرد با آن‌ها رفتار می‌شود.

این مفهوم را با ارائه مثالی در ادامه بیشتر توضیح می‌دهیم؛ اما نخست در مورد کلاس‌ها صحبت می‌کنیم. کلاس‌ها نوعی از اشیای مقدماتی هستند. آن‌ها فهرستی از خصوصیات دارند که وقتی تعریف می‌شوند، تبدیل به یک شیء می‌گردند.

برای مثال به یک بازی شطرنج اشاره می‌کنیم. در بازی شطرنج می‌توانیم کلاسی به نام مهره (Piece) داشته باشیم. درون این کلاس مهره، فهرستی از خصوصیات را به صورت زیر داریم:

- رنگ
- ارتفاع
- شکل
- حرکت‌های مجاز

هر شیئی صرفاً یک وهله خاص از چیزی که به آن کلاس تعلق دارد را تعریف می‌کند. بنابراین می‌توانیم شیئی به نام وزیر سفید داشته باشیم. این شیء می‌تواند تعاریفی برای هر چهار خصوصیت خود داشته باشد (رنگ: سفید، ارتفاع: بلند، شکل: استوانه‌ای دارای تاج، حرکت: به تعداد نامحدود در هر جهت) این شیء می‌تواند متدها یا تابع‌هایی نیز داشته باشد.

۲-۳-۱) اصول برنامه نویسی شی گرا :

سؤالی که در این مرحله ممکن است بپرسید این است که چه چیزی باعث می‌شود برنامه‌نویسی شیءگرا بهتر از برنامه‌نویسی رویه‌ای باشد؟ به طور خلاصه باید گفت که زبان‌های برنامه‌نویسی شیءگرا مانند جاوا امکان سازمان‌دهی داده‌ها و کد را به ترتیبی فراهم می‌سازند که در پروژه‌های بزرگ‌تر سازگاری بیشتری داشته باشند. برای این که این مسئله را بیشتر توضیح دهیم در ادامه چهار اصل برنامه‌نویسی شیءگرا را توضیح خواهیم داد.

۲-۳-۱-۱) تجرید در برنامه نویسی شی گرا :

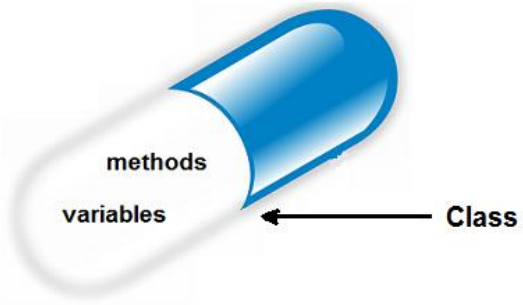
این که قصد دارید از چیزی استفاده کنید، دلیل نمی‌شود که طرز کار آن را می‌دانید. برای مثال یک دستگاه اسپرسو کاملاً پیچیده است، اما شما لازم نیست طرز کار آن را بدانید. شما کافی است بدانید که وقتی دکمه روشن دستگاه را می‌زنید، اسپرسو دریافت خواهید کرد.

همین مسئله در مورد برنامه‌نویسی شیءگرا نیز صدق می‌کند. در مثال بازی شطرنج می‌توانیم یک متد move() داشته باشیم. این متد ممکن است نیازمند مقادیر بالایی از داده‌ها و متدهای دیگر باشد. همچنین ممکن است

نیازمند متغیرهایی به صورت موقعیت اولیه و نهایی مهره باشد. علاوه بر این موارد ممکن است به یک متد دیگر برای زمانی که مهره‌ای یک مهره دیگر را می‌زند نیاز داشته باشد.

اما نیاز نیست که این‌ها را بدانید و همه آنچه که باید بدانید این است که زمانی که از مهره‌ای می‌خواهیم حرکت کند، مهره حرکت می‌کند. این مفهوم تجرید است.

۲-۳-۱-۲) کپسوله سازی در برنامه نویسی شی گرا :



کپسوله‌سازی یکی از روش‌هایی است که برنامه‌نویسی شیءگرا برای ساخت تجرید استفاده می‌کند. هر شیء مجموعه‌ای از داده‌ها است که با آن به عنوان یک واحد مجزا برخورد می‌شود. درون این اشیا نیز داده‌ها به صورت متغیر و متد قرار دارند.

متغیرهای درون شیء به طور کلی مجزا باقی می‌مانند و این به آن معنی است که اشیا و متدها نمی‌توانند به همدیگر دسترسی داشته باشند. اشیا تنها از طریق استفاده از متدهایشان تأثیر می‌پذیرند.

یک شیء وزیر می‌تواند شامل چند بخش از اطلاعات باشد. برای نمونه می‌تواند متغیری به نام «موقعیت» داشته باشد که مکان آن را روی صفحه تعیین می‌کند. این متغیر برای استفاده از متد move() لازم است. و همچنین یک متغیر رنگ نیز دارد.

با اعلان خصوصی متغیر موقعیت و اعلان عمومی متد move() برنامه‌نویس می‌تواند از تأثیرپذیری متغیرهای حرکت از اشیای دیگر جلوگیری نماید. رنگ مهره نیز اگر به صورت خصوصی اعلان شود، اشیای دیگر نمی‌توانند

آن را تغییر دهند؛ مگر این که متدی اجازه این کار را به آن‌ها بدهد. البته در این مورد چنین کاری مجاز نیست چون رنگ مهره‌های شطرنج همواره ثابت است.

۲-۳-۱-۳) وراثت در برنامه نویسی شی گرا :

زبان‌های برنامه‌نویسی شیءگرا علاوه بر کلاس دارای زیرکلاس نیز هستند. این زیرکلاس‌ها شامل همه خصوصیاتی هستند که کلاس‌های والدشان دارند؛ اما می‌توانند خصوصیات دیگری را نیز بپذیرند.

در مورد مثال بازی شطرنج، مهره‌های پیاده به متدی نیاز دارند که در صورت موفقیت در رسیدن به انتهای صفحه، آن‌ها را به وزیر تبدیل کند. مثلاً این متد را می‌توان transformPiece() نامید.

همه مهره‌ها به چنین متدی نیاز ندارند. بنابراین لازم نیست که این متد را در کلاس مهره‌ها قرار دهیم. بلکه می‌توانیم یک زیرکلاس از کلاس مهره‌ها به نام «پیاده» ایجاد کنیم. از آنجا که زیرکلاس‌ها همه خصوصیات کلاس «مهره» را ارث می‌برند، این وهله از زیرکلاس «پیاده» نیز شامل رنگ، ارتفاع، شکل و حرکت‌های مجاز خواهد بود.

اما علاوه بر خصوصیات فوق یک متد transformPiece() نیز در آن تعریف می‌کنیم. در این حالت لازم نیست که نگران استفاده نادرست از این متد بر روی مهره رخ (قلعه) باشیم.

ایجاد زیرکلاس‌ها باعث صرفه‌جویی زیادی در زمان برنامه‌نویسی می‌شود. به جای ایجاد کلاس‌های جدید برای هر چیز، می‌توان تنها یک کلاس پایه نوشت و سپس آن را هر زمان که نیاز باشد، به زیرکلاس‌های جدید گسترش داد.

۲-۳-۱-۴) چندریختی در برنامه نویسی شی گرا :

چندریختی نتیجه وراثت است. درک کامل چندریختی نیازمند کسب حداقلی از دانش برنامه‌نویسی است و از این رو در این بخش تنها به صورت مقدماتی آن را معرفی می‌کنیم. به طور خلاصه چندریختی به برنامه‌نویس امکان می‌دهد که متدهایی با نام یکسان را بر روی اشیای مختلف استفاده کند.

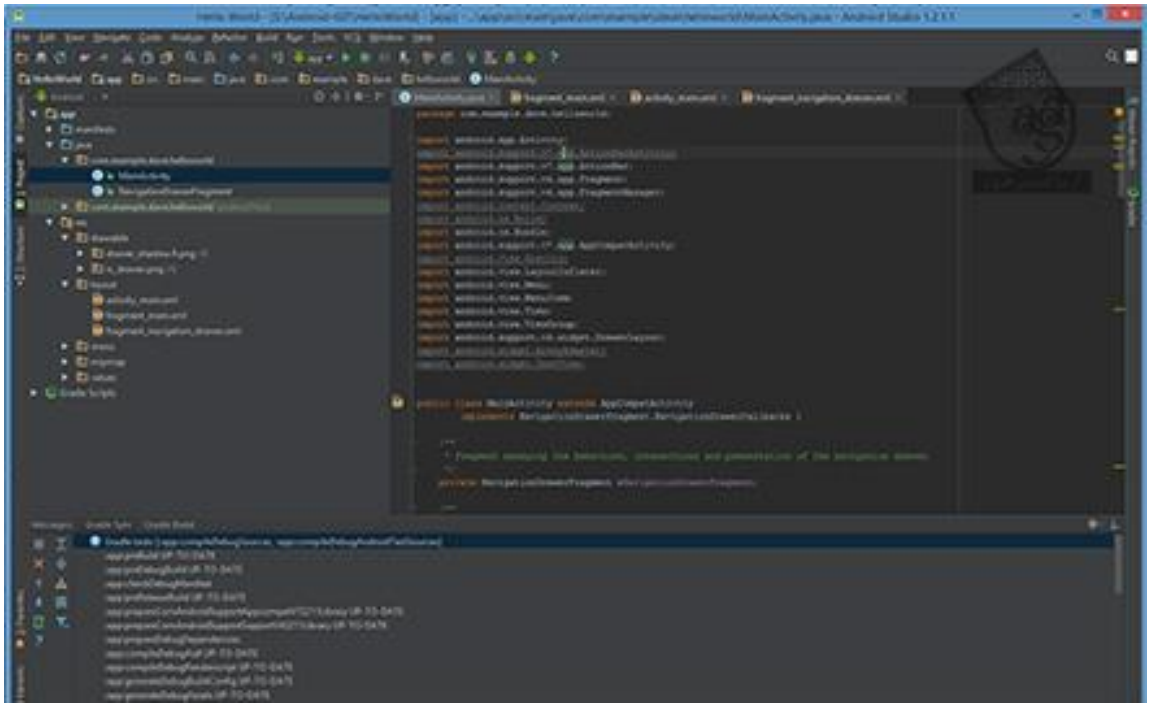
برای نمونه فرض کنید در کلاس «مهره»-ها که در بخش‌های قبلی تعریف کردیم یک متدmove() داشتیم که هر مهره را در همه جهات به اندازه یک واحد حرکت می‌داد. این متد برای مهره شاه مفید است؛ اما فقط برای این

مهره به درد می‌خورد و در مورد مهره‌های دیگر به کار نمی‌آید. برای حل این مشکل می‌توانیم متدmove() جدیدی در زیرکلاس «مهره رخ» تعریف کنیم که به صورت حرکت به تعداد نامحدود در جهت‌های جلو، عقب، چپ و راست باشد.

در این حالت زمانی که برنامه‌نویس متد move() را فراخوانی می‌کند، نوع مهره را نیز به عنوان ورودی (آرگومان) متد ارسال می‌کند تا برنامه بداند که دقیقاً چه مهره‌ای را باید حرکت دهد. بدین ترتیب مقدار زمان زیادی صرفه‌جویی می‌شود. این زمان بدون وجود چندریختی باید صرف این می‌شد که تشخیص دهید چه نوع حرکتی باید انجام دهید.

۲-۴) معرفی محیط Android Studio :

نرم افزار اندروید استودیو (Android Studio) ، یک محیط توسعه یکپارچه (IDE) رسمی برای توسعه پلتفرم و برنامه نویسی اندروید است.
این نرم افزار در ۱۶ می سال ۲۰۱۳ در کنفرانس Google I/O معرفی شد. اندروید استودیو تحت لیسانس Apache License 2.0 به صورت رایگان در دسترس قرار دارد.
اندروید استودیو در می سال ۲۰۱۳ از ورژن ۰,۱ در مرحله پیش نمایش قرار داشت، سپس با شروع ورژن ۰,۸ وارد مرحله بتا شد که در ژوئن سال ۲۰۱۴ عرضه شد. اولین نسخه ثابت آن در دسامبر سال ۲۰۱۴ عرضه شد و با ورژن ۱,۰ شروع شد.
براساس نرم افزار IntelliJ IDEA از شرکت JetBrains ، اندروید استودیو مخصوص توسعه اندروید طراحی شده است. لینک دانلود آن برای ویندوز، Mac OS X و لینوکس قرار دارد و به عنوان IDE اصلی گوگل برای توسعه برنامه های محلی اندروید، جایگزین Eclipse Android Development Tools شده است.



۲-۵) کد های مورد استفاده در برنامه :

۲-۵-۱) کد های Viewmodel :

```
;package com. chartecgmostafa.viewmodel
import android.app.Application;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;

import com.chartecgmostafa.repo.OnReadFileCallback;
```

```
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;

import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.util.List;

public class ChartActivityViewModel extends AndroidViewModel}

    private static final int STATE_FINISHED = 423;
    private static final int STATE_READING = 783;
    private static final int STATE_NOT_READ = 65;

    private AsyncTask task;

    private float scale = 10f;
    private static final int window = 30000;

    private int fileReadState = STATE_NOT_READ;
    private String filename;

    private List<Float> processedSignal;
```

```
    private List<Float> fileSignal;
public ChartActivityViewModel(@NonNull Application application)}
        super(application);

        processedSignal = new ArrayList;()<>
        fileSignal = new ArrayList;()<>
{



    public float getScale} ()
        return scale;
{

    public void setScale(float scale)}
        this.scale = scale;
{



    public void getProcessedSignal(File file, OnReadFileCallback callback)}
        if (!file.exists())}
            Toast.makeText(getApplication(), "Error: Could not Find File..",
Toast.LENGTH_SHORT).show;()
            return;
{

        String filename = file.getAbsolutePath;()
```

```
        switch (fileReadState)}
            case STATE_NOT_READ:
                this.filename = filename;
                fileReadState = STATE_READING;
                task = new ReadFileAsyncTask(this, file, callback).execute;()
                break;
            case STATE_READING:
                if (!filename.equals(this.filename))}
                    this.filename = filename;
                    task.cancel(false);
                    fileSignal.clear;()
                    processedSignal.clear;()
                    task = new ReadFileAsyncTask(this, file, callback).execute;()
{
                break;
            case STATE_FINISHED:
                if (filename.equals(this.filename))}
                    callback.onFileFinished(fileSignal, processedSignal);
{                 else}
                    this.filename = filename;
                    fileReadState = STATE_READING;
                    task.cancel(false);
                    fileSignal.clear;()
                    processedSignal.clear;()
                    task = new ReadFileAsyncTask(this, file, callback).execute;()
{
```

```
            break;
        default:
            try}
                throw new Exception("BUG in Switch cases");
 {          catch (Exception e} (
                e.printStackTrace;()
{
{
{
   public void addEntry(final LineChart chart, int dataSetIndex, List<Float>
newDataList)}
LineData data = chart.getData;()
      if (data != null)}
         ILineDataSet set = data.getDataSetByIndex(dataSetIndex);
  Log.i("gooded", "addEntry: " + newDataList.size() + " , " + "");
         for (Float newData : newDataList)}
            data.addEntry(new Entry(set.getEntryCount()/360f, newData),
dataSetIndex);
{
         chart.setVisibleXRange(0f, scale);
         chart.notifyDataSetChanged;()
         chart.invalidate;()
{
{
   private static class ReadFileAsyncTask extends AsyncTask<Void, List<Float>,
Void} <
      private ChartActivityViewModel vm;
```

```
    private File file;
    private OnReadFileCallback onReadFileCallback;
    private ReadFileAsyncTask(ChartActivityViewModel vm, File file,
OnReadFileCallback onReadFileCallback)}
        this.vm = vm;
        this.file = file;
        this.onReadFileCallback = onReadFileCallback;
{
@       Override
    protected Void doInBackground(Void... voids)}
        try}
            RandomAccessFile filePtr = new RandomAccessFile(file, "r");

            int n = 0;
            List<Float> y0 = new ArrayList;()<>
            List<Float> y1 = new ArrayList;()<>
            List<Float> y2 = new ArrayList;()<>
            filePtr.seek;(٠)
            long fileSize = filePtr.length() - 3;
            while (filePtr.getFilePointer() < fileSize)}
                byte[] b = new byte;[٣]
                int[] i = new int;[٣]
                filePtr.read(b);
                i[0] = b[0] & 0x00ff;
                i[1] = b[1] & 0x00ff;
                i[2] = b[2] & 0x00ff;
```

```
            float num = ((i[1] & 0x000f) << 8) | i;[ٔ]


            vm.fileSignal.add(num);
            if (n >= 4)}
               y0.add(n - 4, Math.abs(vm.fileSignal.get(n) - vm.fileSignal.get(n -
2)));
               y1.add(n - 4, Math.abs(vm.fileSignal.get(n) - 2 * vm.fileSignal.get(n
- 2) + vm.fileSignal.get(n - 4)));
               y2.add(n - 4, 1.3f * y0.get(n - 4) + 1.1f * y1.get(n - 4));
               if (n >= 11)}
                  float sum = 0;
                  for (int k = 0; k <= 7; k++)}
                     sum += y2.get(n - 4 - k);
{
                  float res = 0.125f * sum;
                  vm.processedSignal.add(n - 11, res);
//                  TODO test:: write way in producer thread consumer thread
way::
                  if (n % window == 0)}
                     List subSignal = makeNewSubList(vm.fileSignal.subList(11 +
n - window, n));
                     List subProcessed =
makeNewSubList(vm.processedSignal.subList(n - window, n - 11));
                     publishProgress(subSignal, subProcessed);
{
{
{
```

```
            n;++
            if (n > 90000) { //TODO
                break;
{


{
 {          catch (IOException e} (
             e.printStackTrace;()
{
        return null;
{
    private List makeNewSubList(List list)}
        List subList = new ArrayList;()
        for (Object o : list)}
            subList.add(o);
{
        return subList;
{


@       Override
    protected final void onProgressUpdate(List<Float>... values)}
        List<Float> signal = values;[٠]
        List<Float> processed = values;[١]
        onReadFileCallback.onReadingFile(signal, processed);
{
@       Override
```

```
        protected void onPostExecute(Void aVoid)}
            onReadFileCallback.onFileFinished(vm.fileSignal, vm.processedSignal);
            vm.fileReadState = STATE_FINISHED;
{
{
{
```

۲-۵-۲) کد های رابط کاربری (UI) :

```
package.chartecgmostafa.ui.listeners;

import android.os.Handler;
import android.view.View;

public abstract class OnDoubleClickListener implements View.OnClickListener}

   private boolean isClicked = false;

   public abstract void onDoubleClicked(View v);

@    Override
   public void onClick(View v)}
      if (!isClicked)}
         isClicked = true;
         new Handler().postDelayed(new Runnable} ()
@             Override
            public void run} ()
               isClicked = false;
```

```
{
;(٣٠٠ ,{
 {       else}
          isClicked = false;
          onDoubleClicked(v);
{
{
{
package com. chartecgmostafa.ui.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.chartecgmostafa.R;
import com.chartecgmostafa.ui.activities.BrowsActivity;
import com.chartecgmostafa.ui.activities.ChartActivity;
```

```
import com.chartecgmostafa.ui.adapters.FileAdapter;


import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;


public class BrowsFragment extends Fragment}


@   Nullable
@   Override
   public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState)}
       View view =
LayoutInflater.from(getContext()).inflate(R.layout.layout_fragment_brows,
container, false);
       initViews(view);
       return view;
{
   private void initViews(View view)}
       TextView tvEmptyFolder = view.findViewById(R.id.tv_folder_empty);
       RecyclerView recyclerView = view.findViewById(R.id.recycler_view);
       String path = getArguments().getString("path");
       File file = new File(path);
       if(filterFiles(file.listFiles()).length == 0)
          tvEmptyFolder.setVisibility(View.VISIBLE);
```

```
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext(),
RecyclerView.VERTICAL, false));
        recyclerView.setAdapter(new FileAdapter(getContext(),
filterFiles(file.listFiles()), new FileAdapter.OnItemClickListener} ()
@           Override
          public void onClick(File file)}
             if(file.isDirectory())
BrowsActivity.addFragmentToStack(BrowsFragment.newInstance(file.getPath()));
             else}
                Intent intent = new Intent(getContext(), ChartActivity.class);
                intent.putExtra("path", file.getAbsolutePath());
                startActivity(intent);
{
{
;(({
{
   private File[] filterFiles(File[] listFiles)}
      List<File> files = new ArrayList;()<>
      for(File file: listFiles)}
         if(!file.isFile())
            files.add(file);
         else if(file.getName().endsWith(".dat"))
            files.add(file);
{
      File[] fileArray = new File[files.size()];
      return files.toArray(fileArray);
{
```

```
    private ArrayList<File> fakeData} ()
        return null;
{
    public static BrowsFragment newInstance(String path)}
        Bundle args = new Bundle;()
        args.putString("path", path);
        BrowsFragment fragment = new BrowsFragment;()
        fragment.setArguments(args);
        return fragment;
{
{
package com.chartecgmostafa.ui.adapters;
import android.content.Context;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.chartecgmostafa.R;
import java.io.File;
public class FileAdapter extends
RecyclerView.Adapter<FileAdapter.FileViewHolder} <
```

```
    private File[] files;
    private LayoutInflater inflater;
    private Context context;
    private OnItemClickListener onItemClickListener;
    public FileAdapter(Context context, File[] files, OnItemClickListener
onItemClickListener)}
        this.files = files;
        this.context = context;
        this.onItemClickListener = onItemClickListener;
{
@    NonNull
@    Override
    public FileViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType)}
        if (inflater == null)}
            inflater = LayoutInflater.from(context);
{
        return new FileViewHolder(inflater.inflate(R.layout.item_file, parent, false));
{


@    Override
    public void onBindViewHolder(@NonNull FileViewHolder holder, int position)
}
        File file = files[position];
        if(file.isFile())}
            holder.ivType.setImageResource(R.drawable.ic_file);
            holder.ivType.setColorFilter(Color.RED);
```

```
{
      else}
         holder.ivType.setImageResource(R.drawable.ic_folder);
         holder.ivType.setColorFilter(Color.BLACK);
{
      holder.tvFilename.setText(file.getName());
{
@    Override
   public int getItemCount} ()
      return files.length;
{
   class FileViewHolder extends RecyclerView.ViewHolder}
      private TextView tvFilename;
      private ImageView ivType;
      FileViewHolder(@NonNull View view)}
         super(view);
         tvFilename = view.findViewById(R.id.tv_filename);
         ivType = view.findViewById(R.id.iv_file_folder);
         view.setOnClickListener(new View.OnClickListener} ()
@              Override
            public void onClick(View v)}
               if(onItemClickListener != null)
                  onItemClickListener.onClick(files[getAdapterPosition()]);
{
;({
{
```

```
{
   public interface OnItemClickListener}
      void onClick(File file);
{
{
package com. chartecgmostafa.ui.activities;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.ProgressBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProviders;
import com.chartecgmostafa.R;
import com.chartecgmostafa.repo.OnReadFileCallback;
import com.chartecgmostafa.ui.listeners.OnDoubleClickListener;
import com.chartecgmostafa.viewmodel.ChartActivityViewModel;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
public class ChartActivity extends AppCompatActivity}
```

```
    private static final String TAG = "goodgood;"
    private LineChart chart;
    private ChartActivityViewModel vm;
@   Override
    protected void onCreate(Bundle savedInstanceState)}
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chart);
        if (getActionBar() != null)
            getActionBar().hide;()
        String path = getIntent().getStringExtra("path");
        File file = new File(path);
        Log.i(TAG, "onCreate: " + file.getAbsolutePath() + " , " + file.exists());
        chart = findViewById(R.id.chartLayout);
        ImageView ivZoomIn = findViewById(R.id.btn_zoom_in);
        ImageView ivZoomOut = findViewById(R.id.btn_zoom_out);
        ivZoomIn.setOnClickListener(new View.OnClickListener} ()
@           Override
           public void onClick(View v)}
               vm.setScale(vm.getScale() / 2f);
               chart.setVisibleXRange(0, vm.getScale());
               chart.invalidate;()
               Log.i(TAG, "onClick: " + vm.getScale());
{
;({
        ivZoomOut.setOnClickListener(new View.OnClickListener} ()
@           Override
```

```
        public void onClick(View v)}
            float target = (chart.getXChartMax() - chart.getXChartMin()) / 2 -
vm.getScale;()
            vm.setScale(vm.getScale() * 2f);
            chart.setVisibleXRange(0, vm.getScale());
            chart.moveViewToX(target);
            chart.invalidate;()
            Log.i(TAG, "onClick: " + vm.getScale());
{
;({
    drawChart(new ArrayList<Float>(), new ArrayList<Float>());
    final ProgressBar progressBar = findViewById(R.id.progress_bar);
    progressBar.setVisibility(View.VISIBLE);
    vm = ViewModelProviders.of(this).get(ChartActivityViewModel.class);
    vm.getProcessedSignal(file, new OnReadFileCallback} ()
@       Override
        public void onReadingFile(List<Float> newSignal, List<Float>
newProcessedSignal)}
            vm.addEntry(chart, 0, newSignal);
            vm.addEntry(chart, 1, newProcessedSignal);
{
@       Override
        public void onFileFinished(List<Float> signal, List<Float>
processedSignal)}
  //            drawChart(signal, processedSignal); //TODO test purpose
            progressBar.setVisibility(View.GONE);
{
```

```
;({
{
   private void drawChart(List<Float> signal, List<Float> processed)}
      List<Entry> ecgEntries = new ArrayList;()<>
      List<Entry> processedEntries = new ArrayList;()<>
      float i = 0;
      for (int n = 11; n < signal.size(); n++)}
         float x = i / 360f;
         ecgEntries.add(new Entry(x, signal.get(n - 11)));
         processedEntries.add(new Entry(x, processed.get(n - 11)));
         i;++
{
      final LineDataSet ecgDataSet = new LineDataSet(ecgEntries, "ECG Data");
      final LineDataSet processedDataSet = new LineDataSet(processedEntries,
"Processed");
      ecgDataSet.setDrawCircles(false);
      ecgDataSet.setCircleColor(Color.BLUE);
      ecgDataSet.setLineWidth(3f);
      ecgDataSet.setDrawValues(false);
      ecgDataSet.setColor(Color.GREEN);
      processedDataSet.setDrawCircles(false);
      processedDataSet.setCircleColor(Color.BLACK);
      processedDataSet.setLineWidth(3f);
      processedDataSet.setDrawValues(false);
      processedDataSet.setColor(Color.RED);
      LineData lineData = new LineData;()
```

```
        lineData.addDataSet(ecgDataSet);
        lineData.addDataSet(processedDataSet);
        chart.setData(lineData);
        chart.setDoubleTapToZoomEnabled(false);
        chart.invalidate;()
        chart.setOnClickListener(new OnDoubleClickListener} ()
@           Override
          public void onDoubleClicked(View v)}
             ecgDataSet.setDrawCircles(!ecgDataSet.isDrawCirclesEnabled());
             ecgDataSet.setDrawValues(!ecgDataSet.isDrawValuesEnabled());

processedDataSet.setDrawCircles(!processedDataSet.isDrawCirclesEnabled());

processedDataSet.setDrawValues(!processedDataSet.isDrawValuesEnabled());
{
;({
{
package chartecgmostafa.ui.activities;
import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import com.chartecgmostafa.R;
import com.chartecgmostafa.ui.fragments.BrowsFragment;
import java.util.Stack;
public class BrowsActivity extends AppCompatActivity}
   private static final int REQ_ACCESS_EXTERNAL_STORAGE = 123;
   private static Stack<Fragment> fragmentStack;
   private static FragmentManager fm;
   public static void addFragmentToStack(Fragment fragment)}
      fm.beginTransaction()
.            hide(fragmentStack.peek())
.            add(R.id.brows_frame, fragment)
.            commit;()
      fragmentStack.push(fragment);
{
@   Override
   protected void onCreate(Bundle savedInstanceState)}
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_brows);
      fragmentStack = new Stack;()<>
      fm = getSupportFragmentManager;()
```

```
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED)}

 //          Permission is not granted

 //          Should we show an explanation?

          if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.READ_EXTERNAL_STORAGE))}

//              TODO inja yaani in ke ghablan ye bar deny karde pass mishe ye payam
behesh neshun dad ke chera permission ro mikhahim

//              TODO momkene havasesh nabude bashe zade bashe always deny,
mishe ye peygham behesh dad ke az to setting dorostesh kon va ye intent zad be
setting

             ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
REQ_ACCESS_EXTERNAL_STORAGE);

 {          else}

 //             No explanation needed; request the permission

             ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
REQ_ACCESS_EXTERNAL_STORAGE);

{

 {       else { // Permission has already been granted

          initFragments;()

{

{

@    Override

   public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults)}

       switch (requestCode)}
```

```
        case REQ_ACCESS_EXTERNAL_STORAGE} :
//              If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)}
                initFragments;()
{           else}

                String appName = getResources().getString(R.string.app_name);
                Toast.makeText(this, appName + " needs this feature to read ECG dat
files from your Storage", Toast.LENGTH_LONG).show;()
                new Handler().postDelayed(new Runnable} ()
@                   Override
                  public void run} ()
                    finish;()
{
;(٣٠٠٠ ,{
{
{
{
{
   private void initFragments} ()
      String rootDir =
Environment.getExternalStorageDirectory().getAbsolutePath;()
      BrowsFragment browsFragment = BrowsFragment.newInstance(rootDir);
      fm.beginTransaction()
.             add(R.id.brows_frame, browsFragment)
.             commit;()
```

```
        fragmentStack.push(browsFragment);
{
@    Override
    public void onBackPressed} ()
        if (fragmentStack.size() <= 1)
            finish;()
        else
            fm.beginTransaction()
.                   remove(fragmentStack.pop())
.                   show(fragmentStack.peek())
.                   commit;()
{
package com.chartecgmostafa.repo.Model;
public class Point}
    private float x;
    private float y;
    public Point(float x, float y)}
        this.x = x;
        this.y = y;
{
    public float getX} ()
        return x;
{
    public float getY} ()
        return y;
{
```

```
{
package com.chartecgmostafa.repo;
import com.chartecgmostafa.repo.Model.Point;
import java.util.List;
public interface OnReadFileCallback}
   void onReadingFile(List<Point> newSignal, List<Point> newProcessedSignal,
List<Point> line);
   void onFileFinished(List<Point> signal, List<Point> processedSignal,
List<Point> line);
{
package com.chartecgmostafa.repo;


public interface OnReadECGFileCallback}
   void onReadNumber(float num1, float num2);
{
package com.chartecgmostafa;
public class Configs}
   public static final float SAMPLING_FREQUENCY = 360f;
//    private static final int SAMPLING_RESOLUTION = 12;
   public static final int WINDOW = 10000; //Baze haee ke hey load mikonim ru
nemudar
   public static final int VALID_HEARTBEAT_THRESHOLD = 80;
   public static final float MIN_HEARTBEAT_PERIOD_SECOND = 0.07f;
//    private static final int MIN_HEARTBEAT_PERIOD_SAMPLES =
(int)(MIN_HEARTBEAT_PERIOD_SECOND * SAMPLING_FREQUENCY);
   public static final float NORMAL_HEARTBEAT_DISTANCE_TOLERANCE
= 0.1f;
```

فصل سوم :

پیشنهادات و منابع

۳-۱) پیشنهادات :

در این پروژه فقط بر روی سه نوع خاص از ناهنجاری های قلبی که تشخیص آنها بر اساس فواصل بین قله های موج QRS استوار بود بحث شد. می توان با استفاده از ویژگی های خاص دامنه QRS و همچنین امواج P و T و همچنین فواصل دیگر بین موج ها و بالطبع تعریف الگوریتم مناسب برای هر کئام، ناهنجاری های بیشتری را شناسایی نمود.

۳-۲) منابع:

1) Rangaraj M Rangayyan, Knovel (Firm)-Biomedical signal analysis_ a case-study approach-IEEE Press _ New York, N.Y. (2002) –opt-red

2) MIT – BIH Arrhythmia Database

۳)پردازش سیگنال های قلبی و تشخیص ناهنجاری به همراه ارسال پیام هشدار برای بیمار و پزشک معالج از طریق سرویس پیام کوتاه ، حامد رحمانی ، مجموعه پروژه های دانشگاه صنعتی شریف

4) "What Is Arrhythmia?". National Heart, Lung, and Blood Institute. July 1, 2011

5) https://aryagostarafzar.com/articles/android-studio/

6) https://www.lydaweb.com

7) https://blog.faradars.org/object-oriented-programming-explained/

8) https://mizbanfa.net