مهندسی نرم‌افزار ۱ - تمرین هشتم

گروه ۲
سیدمهدی حاجی سیدحسین - امیرعلی شهریاری - علیرضا حسینی
810100125 - 810100173 - 810100118

شناسه کامیت حاوی بوی بد	شرح مختصر بوی بد	شناسه کامیت ریفکتور شده	لینک مربوط به smell	توضیحات
83696a4	طولانی بودن متد validate کردن order ها	5ca3657	https://refactoring.guru/smells/long-method	دو متد جدا یکی برای order های عادی و دیگری برای iceberg درست شد
5ca3657	ایمپورت های اضافی در فانکشن های validation	2c8667d	https://refactoring.guru/smells/long-parameter-list	لزومی نداشت که security به عنوان ورودی به توابع validationداده میشد
3d64bb9	طولانی بودن متد و چند کاره بودن آن	a09ebc5	https://refactoring.guru/smells/long-method	تبدیل به سه متود با وظایف مجزا شد isBetterOpeningPrice بولین شد.
99d9766	وجود کد تکراری قابل تفکیک	68248dc	https://refactoring.guru/smells/duplicate-code	Cast Type کار مشابه میکرد که برای sell & buy مرج شد
8b7fe4d	کد مرده و بلااستفاده	21063a7	https://refactoring.guru/smells/dead-code	این متد از MatchResult حذف گردید
f6b4fd2	Message Chains ها و تو در تویی	cf29061	https://refactoring.guru/smells/message-chains	متد به دو تابع دیگر شکست که برای بررسی ولید بودن و محاسبه tradableQuantity از شکل قبلی بهبود میابد.
854bef2	متد با انجام دو کار متفاوت	9135726	https://refactoring.guru/smells/long-method	به توابع کوچک تر بر انجام اعمالشان تجزیه شد.
b8d6d18	مشکل جدی در تست	decac6c		iceberg_order_behaves_normally_before_being_queued در این تست matcher مشکل داشت چون که ما در ریفکتوری که داشتیم به mather یک cotroller اضافه کردیم و به این مشکل برخوردیم که در تست macher درست در تست ها وایر نشده بود . (میتواند به خاطر derty context هم باشد)

شناسه کامیت حاوی بوی بد	شرح مختصر بوی بد	شناسه کامیت ریفکتور شده	لینک مربوط به smell	توضیحات
	حذف کامنت ها	8de491d	https://refactoring.guru/smells/comments	حذف کامنت های بلا استفاده در کل پروژه
	حذف ایمپورت بی استفاده	fe9e885	https://refactoring.guru/smells/dead-code	حذف کلیه ایمپورت های بدون استفاده از کل پروژه برای حذف کد های مرده

ایده های بهبود :

Dependency Inversion : جداسازی تابع match در Matcher و تقسیم وظایف آن به ماژول های مجزا برای هر بخش ,Credit Ownership, MinimumExecution به صورتی که به پیاده سازی وابسته نبوده و دو لایه متفاوت از طریف یک Interface به یکدیگر مرتبط باشند. (بخشی از مثل Credit Control در کامیت های d4c329b و بعد از آن پیاده سازی شده است)

Single Responsibility Principle : مشخصا کلاس هایی در این برنامه وجود دارند که بیش از یک کار انجام میدهند. برای مثلا Security به بخشی از اعتبار و ... رسیدگی می کند که لزوما به خود security مربوط نیست. با جداسازی این دست از وظایف از یکدیگر از coupling آنها جلوگیری کرد

Notification Design Pattern : در کلاس security میشد در متد های newOrder , updateOrder از دیزاین پترن notification استفاده کرد مانند OrderHandler تمام پیام ها و یا حتی اخطار ها را در یک List<String> errors ذخیره کرد و در نهایت همه را باهم پابلیش کرد .