سپهر خسته

> همون طور که میدونیم سپهر یکمی دست و پا چلفتیه!!

واسه همین ممکنه ورودی های غلط به برنامه ای که برای تاریخ و زمان نوشته بده! از طرفی دیگه از دیباگ کردن کداش و assert نوشتن خسته شده! حالا سپهر به کمک شما احتیاج داره که اگه ورودی هاشو غلط داد بتونه هندلشون کنه.

حالا مشکلاتی که میتونن پیش بیان به شکل زیر هستن:

کلاس `Time`

- در صورت بروز خطا در متغیر **دقیقه یا ساعت**، یک شیئ از کلاس `TimeException` را `throw` کنید که شامل یک متغیر برای نمایش مقدار خطا و یک متن برای توضیح بیشتر خطاست.

کلاس `Date`

- در صورت بروز خطا برای **روز**، مقدار `int` را `throw` کنید.
- اگر برای مقدار **ماه** خطا پیش اومد، `out_of_range` را `throw` کنید. (برای اطلاعات بیشتر درمورد این کلاس، به این لینک مراجعه کنید.)
- در صورت بروز خطا برای **سال** یک شیئ از کلاس `DateException` را `throw` کنید که مشابه کلاس `TimeException` است.

کلاس `DateTime`

- خطاهای مربوط به متغیر **دقیقه** کلاس `Time` را رسیدگی کنید و نزدیک ترین عدد را کنید.
- خطای `int` مربوط به **روز** در کلاس `Date` را رسیدگی کنید و نزدیک ترین عدد را جایگزین کنید.
- خطا های مربوط به **ساعت و یا سال** را `catch` کنید و خطای `DateTimeException` را `throw` کنید. (چون خطای این دو مورد مهم هستند و امکان رسیدگی به آن در این کلاس نیست.)

موارد مهم:

- در برنامه خود تا حد امکان از **تکرار کد خودداری کنید** و تا حد امکان خطا ها را در یک قسمت رسیدگی کنید.
- کلاس های خطای شما باید از کلاس `Exception` ارث ببرند.(توضیحات این کلاس در این لینک است.)
- در `main` برنامه خطای مربوط به `DateTimeException` را `catch` کنید و در صفحه کنسول متن مربوط به خطا را چاپ کنید.
- در `main` برنامه مثال هایی برای **چک کردن سناریو های بالا** بزنید.

دست گرمی پروژه :)

در این سوال شما باید با توجه به فایل پروژه ای که مربوط به پروژه های قبلی این درس بوده است فایل PDF شامل یکی از پروژه های درس برنامه نویسی پیشرفته است. در این سوال شما باید با توجه به توضیحات پروژه UML این پروژه را طراحی کنید.

در نهایت یک فایل PDF شامل طراحی خود به همراه لینک یک **ویدیو حداکثر ۳ دقیقه** ای برای توضیح و دفاع از طراحی خود آپلود کنید.

در هنگام طراحی به موارد زیر توجه کنید:

- به خوبی از روابط شیئ گرایی استفاده کنید و تا حد امکان از اصول **SOLID** که در کارگاه بحث شد استفاده کنید. (برای اطلاعات بیشتر می‌توانید به این لینک مراجعه کنید.)
- نیازی به جزئیات کامل نیست و توابع مهم در طراحی کافی است.
- به دسترسی های هر کلاس با توجه به توابعی که دارد دقت کنید.