☆ ریسورت سرچ ① cpu :

- در این سرچ باید برنامه فهرستی از کارهارو به عنوان ورودی بگیره که

هرکدوم دارای یه زمان پردازش مختص خودشون برای اجرای کامل اون کاره؛

- کاری که cpu باید انجام بده اینه که کاری با زمان پردازش کمتر رو انتخاب کنه و یک thread جدید برا اش ایجاد کنه.
- سه اول باید این سرچ رو بصورت کامل اجرا بکنه بعدش بریم سراغ سرچ های بعدی. (و سرچ main رو join کنیم).
- بعدش وقتی یه سرچ بصورت کامل انجام شده ID اون کار رو به عنوان کارهای انجام شده به یه آرایی اضافه میکنیم.

#توضیح راه حل رکدها: اولش یه کلاس Task ایجاد میکنیم و اون رو از روی اینترفیس Runnable ایمپلیمنت میکنیم.
دوتا متغیر ID و process Time ایجاد میکنیم و کانستراکتورش رو مینویسیم.
بعدش متد run() رو override میکنیم و توش یه Try-catch میزاریم برای sleep کردن سرچ به اندازه اون عددی که داره...

مهم‌ترین قسمت برنامه در واقع متد startSimulation هست که یه ArrayList ای از Tasks به عنوان ورودی می‌گیره (که مرتب سازی رو روی اینا انجام بده). بعد یه ArrayList برای کارهای انجام شده می‌سازیم که به محض اینکه یه ترد به طور کامل انجام شد، اون رو بذاریم توی این ArrayList به نام executedTasks.

بعدش یه حلقه می‌زنیم تا زمانی که اون لیستی که به عنوان ورودی گرفتیم خالی نشده، این کارها رو روی Task ها انجام بدیم:

اولش باید کوتاه‌ترین تایم فرآیند Tasks رو پیدا کنیم و براش یه ترد بسازیم و این ترد رو start کنیم؛ بعد باید try-catch اون رو به ترد اصلی هم join می‌کنیم. در آخر هم ID این تردی که کامل اجرا شده رو به executedTasks، add می‌کنیم...

توی کلاس main، task های دلخواهمون رو ایجاد می‌کنیم و بعدش متد startSimulation رو روی کلاس جدیدی که ساختیم، اجرا می‌کنیم. (به عنوان ورودی هم این task هایی که ایجاد کردیم رو میدیم).

و هر در آخر برنامه رو که run کنیم، به ترتیب processing Time شون، ID ترد ها باید باشه... ✓

★ ریپورت تمرین ۲

- در این تمرین باید برنامه ای می‌نوشتیم که تمام اعداد صحیح بین 1 تا عدد ورودی که کاربر میده رو که بر ۳، ۵ یا ۷ بخش‌پذیر هست رو پیدا کنه و جمعشون رو return کنه. باید برای حل تمرین از thread ها استفاده می‌کردیم.

- اولش یه کلاس Task ایجاد می‌کنیم و اون از روی اینترفیس Runnable implement می‌کنیم. بعد دوتا متغیر n به عنوان آخرین بازه و result به عنوان لیستی که یک اعداد بیت لونده رو بخش‌پذیر (ذخیره کرده) درست می‌کنیم و کانستراکتورش رو هم ایجاد می‌کنیم. بعدش متد run رو override می‌کنیم و توش یه حلقه می‌زنیم که اعداد 1 تا n رو بررسی کنه و اگه به هرکدوم از اعداد ۳، ۵ یا ۷ بخش‌پذیر بود، اد کنه به result.

- مرحله بعد میرسیم به اصلی‌ترین قسمت برنامه که متد getSum هستش. اولش دوتا متغیر sum=0 و threadCount=۳ ایجاد می‌کنیم همچنین برای دوتا ArrayList به اسم‌های threads و result جا در نظر می‌گیریم. متغیر chunkSize رو تعریف می‌کنیم که برابره با n تقسیم بر تعداد ترد ها (۳). و اون ... رو برابر end می‌ذاریم. توی این متد یه حلقه هم می‌زنیم تا اعداد تکراری شمرده نشن در نهایت و بعدش ترد رو start می‌کنیم. سپس با یه try...catch، اون‌ها رو به ترد اصلی join می‌کنیم. توی متد main اولش از کاربر n مورد نظر رو دریافت می‌کنیم و تابع getSum رو روش اجرا می‌کنیم و در نهایت sum پرینت می‌شه.