استاد: **رامتین خسروی**
طراحان: **بردیا اقبالی**، **امیرحسین حبیب‌وند**
مهلت تحویل: جمعه ۱۶ فروردین ۱۳۹۸، ساعت ۲۳:۵۵

۱ بازسازی

تعاریف زیادی از "کد تمیز"[۱] وجود دارد؛ اما احتمالا یکی از بهترین تعریف‌ها متعلق به "بیارنه استراستروپ" [۲] خالق و توسعه‌دهنده‌ی زبان C++ است. وی در تعریف خود از یک کد تمیز، دو مورد زیر را به عنوان معیارهای اساسی یک کد تمیز بر می‌شمارد:

- منطق و الگوریتم کد باید آن‌قدر واضح و قابل‌فهم باشد که اشکالات و نقص‌های جزئی نتوانند از چشم برنامه‌نویس و آزمونگر کد دور بمانند؛ ضمن این که وضوح کد باید به حدی بالا باشد که برنامه‌نویس را از نوشتن کامنت[۳] بی‌نیاز کند.
- کارایی[۴] برنامه نوشته‌شده باید در بهینه‌ترین[۵] شکل ممکن باشد تا بعدها برنامه‌نویس دیگری به بهانه‌ی بهینه‌سازی[۶] برنامه‌ی سابق با ایجاد تغییرات نادرست سبب نامنظم‌شدن و کثیف‌شدن کد نشود.

در عمل، در اکثر مواقع شما بعد از یک طراحی نسبتاً خوب و پیاده‌سازی آن، برای مدتی طولانی از آن کد برا هدف خود استفاده می‌کنید و در طول این مدت تغییرات و قابلیت‌های زیادی را به آن می‌افزایید.

پس از مدتی نه‌چندان طولانی، این تغییرات باعث می‌شوند که شما دیگر عملکرد کد را به‌وضوح متوجه نشوید و به تبع آن، توانایی تغییر و ارتقای کد را نیز از دست می‌دهید. همین زنجیره‌ی اتفاقات به ظاهر ساده در تاریخچه‌ی نسبتاً کوتاه توسعه‌ی نرم‌افزاری باعث نابودشدن شرکت‌های بسیاری در این عرصه شده است.

حال با توجه به خطرات و مشکلاتی که یک کد کثیف به همراه دارد، باید راه‌حلی برای رفع کثیف‌بودن کد و جلوگیری از ایجاد آن ارائه دهیم. شما در این تمرین کامپیوتری با روند بازسازی[۷] کد آشنا می‌شوید.

بازسازی عملیاتی است که در طی آن ساختار یک نرم افزار به صورتی تغییر و بهبود می‌یابد که بدون از دست‌رفتن کارآیی‌ها و تغییر رابط کاربری[۸] برنامه، ساختار درونی کد به طرز قابل توجهی تمیزتر می‌شود.

بنیادی‌ترین مفهوم یاری‌کننده‌ی یک برنامه‌نویس در طی عملیات بازسازی شناخت عناصری است که باعث کثیف شدن کدها می‌شوند و به اصطلاح به آن‌ها Smells Code گفته می‌شود.

وظیفه‌ی شما در این تمرین بازسازی کد خودتان در تمرین اول درس است؛ بنابراین **خوانایی و تمیزبودن کد** در این تمرین بیشترین اهمیت را دارد. در ادامه توضیحاتی درباره‌ی بازسازی کد ارائه می‌شود. پیشنهاد می‌کنیم که ابتدا تا پایان صورت تمرین را مطالعه کنید و سپس شروع به بازسازی کد خود کنید.

[1] Clean Code
[2] Bjarne Stroustrup
[3] Comment
[4] Performance
[5] Optimal
[6] Optimization
[7] Refactoring
[8] Interface

۲ کد تمیز

عواملی در کد وجود دارند که ممکن است باعث کثیف‌شدن آن شوند؛ در ادامه برخی از این عوامل توضیح داده شده‌اند. توجه کنید که در انتهای این تمرین نمره شما فقط بر اساس عوامل زیر سنجیده می‌شود و به ازای هر یک از موارد زیر که در کد شما وجو داشته باشد نمره‌ی شما کاسته خواهد شد. ساختار کلی کد و طراحی شما نباید تغییر کند و فقط ساختار درونی آن می‌تواند تغییر کند.
این عوامل خلاصه‌ای از کتاب Code Clean هستند و عبارت انتهای هر عامل فصل و شماره‌ای آن عامل را در کتاب نشان می‌دهد.

۱.۲ نام‌گذاری (فصل ۲)

- نام کلاس‌ها و اشیا[۹] باید عبارت‌های اسمی[۱۰] مثل Customer و Account باشند و با حرف بزرگ[۱۱] شروع شوند.
- نام توابع باید عبارت‌های فعلی[۱۲] مثل get ، set و deletePage باشند و با حرف کوچک شروع شوند.

۲.۲ توابع (فصل ۳)

- Functions should do one thing. They should do it well. They should do it only.
- توابع باید تا حد امکان **کوتاه** باشند. تابع باید حداکثر ۶ (؟۸) خط باشد.
- تابع باید حداکثر یک به یک سطح پایین‌تر دسترسی داشته باشد. مثلا با یک حلقه روی لیستی از اشیا و تغییر ویژگی[۱۳]های هر کدام از اشیا دسترسی به ۲ سطح پایین‌تر از تابع است. و این عملیات باید در تابعی جداگانه پیاده‌سازی شود.
- تعداد آرگومان‌های تابع تا حد امکان کم باشد(حداکثر ۳ تا). در صورت امکان از آرگومان‌هایی از نوع اشیا استفاده شود. مثلا به جای دو متغیر از نوع double از یک شئ از نوع point استفاده کنیم.

۳.۲ فرمتینگ (فصل ۵)

- **دندانه‌گذاری**[۱۴] در کد اهمیت بالایی دارد و حتما هر محدوده[۱۵] باید یک دندانه داخل‌تر باشد. همچنین هر تابع باید حداکثر یک یا دو دندانه داخل رفته باشد.
- در نام‌گذاری توابع و متغیرها از یک روش واحد نام‌گذاری[۱۶] استفاده شده باشد. مثلا یا همه متغیرها به صورت Camel-Case نام‌گذاری شده باشند و یا همه به صورت Snake_case باشند. در هر صورت دیگر قوانین نام‌گذاری باید رعایت شوند.

[9] Objects
[10] Noun Phrase
[11] Capital
[12] Verb Phrase
[13] property
[14] Indentation
[15] Scope
[16] Naming Convention

۴.۲ Smells and Heuristics (فصل ۱۷)

۱.۴.۲ کامنت‌ها

- در این تمرین کامنت‌گذاری به هیچ نحوی قابل قبول نیست.

۲.۴.۲ توابع

- آرگومان‌هایی که به عنوان خروجی تابع استفاده می‌شوند. یک تابع فقط باید بتواند از طریق مقدار بازگشتی خود بر محیط بیرون تأثیر بگذارد و نتواند از طریق تغییر آرگومان‌ها بر محیط تأثیری داشته باشد.
- آرگومان از نوع بولین[۱۷] برای تعیین نحوه عملکرد کد؛ مثلا پاس دادن یک متغیر به نام flag به تابع، فقط برای اجرای یک بخش کد در حالتی خاص. چنین تابعی در واقع دو تابع مختلف است که باید به صورت جدا از هم پیاده‌سازی شوند و در زمان مناسب صدا[۱۸] شوند.

[17] Boolean
[18] Call