**پروژه ۱ کامپایلر**

هدف از پروژه‌های کامپایلر، پیاده‌سازی اجزاء مختلف یک کامپایلر واقعی برای نسخهٔ محدود شده‌ای از زبان جاوا (که به آن زبان **میوا** -مخفف مینیمال جاوا- می‌گوییم) می‌باشد. سینتکس زبان میوا کاملا مشابه با زبان جاوا می‌باشد که در آن موارد زیر باید حتما وجود داشته باشند:

* امکان تعریف تابع main (بدون آرگومان ورودی)
  + تابع main تنها تابع قابل تعریف در زبان میوا می‌باشد و مابقی کد در داخل تابع main قرار می‌گیرد.
* پشتیبانی از تایپ‌های ابتدایی int و boolean برای تعریف متغیر
* امکان تعریف شرط if (هم بدون else و هم با آن)
* پشتیبانی از ثابت‌هایی نظیر true و false و همینطور اعداد صحیح
* پشتیبانی از عملگرهای زیر:
  + تخصیص یا =
  + عملگرهای ریاضی \* / + - به همراه ) (
  + عملگرهای منطقی && || == != ! به همراه <= >= < >

**پروژه اول** در مورد پیاده سازی تحلیل‌گر لغوی کامپایلر برای زبان میوا می‌باشد. برنامه شما یک فایل را از ورودی دریافت کرده و در صورت درست بودن فایل از نظر تحلیل‌گر لغوی، توکن‌های موجود در فایل را به صورت سه‌تایی lexeme class line بازمی‌گرداند که در آن lexeme اشاره به لغت شناسایی شده، class اشاره به کلاسی که لغت به آن تعلق دارد و line اشاره به شمارهٔ سطر برنامه می‌کند. بیشتر کلاسهای موردنیاز برای زبان در کلاس درس توضیح داده شده‌اند (نظیر identifier و keyword و...)، با این وجود ممکن است لازم باشد تا متناسب با قواعد فوق کلاس‌های دیگری نیز به زبان افزوده شود. نام متغیرها نیز از قوانین جاوا تبعیت می‌کنند؛ یعنی با عدد شروع نمی‌شوند و نمادهای خاص مثل # و $ و... را نمی‌گیرند.

**نکته مهم ۱:** اگر در برنامه خطایی از نظر تحلیلگر لغوی وجود داشته باشد، شماره خط به همراه نوع خطای مربوطه باید در خروجی گزارش شود. هر خط ممکن است شامل چند عبارت باشد (مثلا int a=2; int b=3;) و اگر چند خطا در هر سطر وجود دارد، تمام آن‌ها باید گزارش شوند.

**نکته مهم ۲:** کد خود را فقط و فقط در داخل فایل‌های نمونه ارسال شده به گروه (توسط آقای زارع‌زاده) بنویسید و توابع مشخص شده را در داخل آن‌ها پیاده سازی کنید. البته می‌توانید بسته به نیاز خود فایل‌ها، کلاس‌ها و یا توابع دیگری نیز به فایل‌های نمونه اضافه کنید. کلاس اصلی پروژه اول کلاس Lexer می‌باشد که دارای یک تابع tokenize برای استخراج لغات از رشته ورودی است که در داخل یک حلقه به صورت مداوم توکن‌های موجود در ورودی را شناسایی می‌کند. وظیفه اصلی شما پیاده سازی کامل تابع addNextToken برای شناسایی یک توکن از جمله ورودی و اضافه نمودن آن به لیست توکن‌های شناسایی شده تاکنون می‌باشد (همچنین، این تابع اندیس انتهای توکن در رشته ورودی را به عنوان خروجی باز می‌گرداند).