Debugging

Spis treści

Zapiski prowadzącego Karola Rogowskiego i uczestnika Bootcampu Zajavka Bartek Borowczyk aka Samuraj Programowania.

Debuggowanie

Przejdźmy do jednej z ważniejszych umiejętności w programowaniu ogólnie, nie tylko w Javie - szukanie błędów w kodzie. Jeżeli nie byłoby debugowania, to jedyną opcją szukania błędów w programie byłoby drukowanie na ekranie każdego wykonywanego kroku razem z wartościami zmiennych w danym kroku.

W praktyce natomiast robi się inaczej, od tego jest debugger (reprezentowany w Intellij przez ikonkę zielonego robaczka).

Czym jest debuggowanie

Czasem można nazywa to odrobaczaniem, bo z angielskiego bug to robak. Albo odpluskwianie, kto jak woli. Generalnie rzecz nazywając, jest to proces szukania i poprawiania błędów w programie.

W programowaniu można zgrubnie wyróżnić kilka rodzajów błędów:

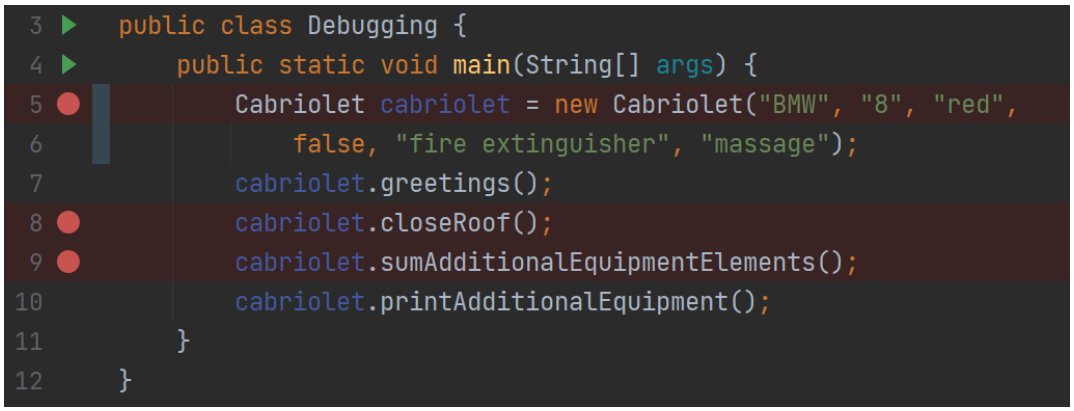
- **błędy na etapie kompilacji** - błąd polegający na problemie w składni, IDE często podpowiada nam co jest nie tak, objawia się tym, że nie możemy poprawnie skompilować kodu,
- **błędy w trakcie wykonywania programu objawiające się wyrzucaniem wyjątków** (o wyjątkach już niedługo) - wtedy na ekranie drukuje się przebieg wywołania programu na czerwono, z zaznaczonymi fragmentami kodu, gdzie wystąpił błąd,
- **błędy w logice programu** (i te są często najgorsze) - program kompiluje się, uruchamia i działa poprawnie w sensie takim, że nie jest wyrzucany żaden z błędów powyżej, ale przykładowo nasz kalkulator liczy, że `2 + 3 = 9`. Żeby dolać oliwy do ognia, to w praktyce takie błędy pojawiają się często w aplikacjach, które są już wdrożone na środowisko produkcyjne i używane na co dzień przez klientów (użytkowników aplikacji). Zdarzają się przypadki, że potrzeba dużo czasu i zabawy, żeby takie błędy znaleźć. Przypadek `2 + 3 = 9` jest prosty, w praktyce zdarzają się o wiele gorsze.

W przypadku gdy potrzebujemy przeprowadzić analizę (nie przestrasz się stwierdzenia, to normalna

rzecz w pracy) występowania błędu, bardzo przydatnym narzędziem staje się **debugger**. Pomaga znaleźć błędy o wiele szybciej niż drukowanie na ekranie wartości po każdym wykonanym kroku. Debugger działa w ten sposób, że możemy określić miejsce w programie, na którym wykonywanie programu ma zostać wstrzymane. Taka pauza. Jesteśmy wtedy w stanie zobaczyć wartość wszystkich zmiennych w danym miejscu w kodzie. Możemy również przechodzić do następnej linijki wywołania programu i widząc, jak zmieniają się wartości zmiennych, przeanalizować sytuację krok po kroku.

Breakpointy

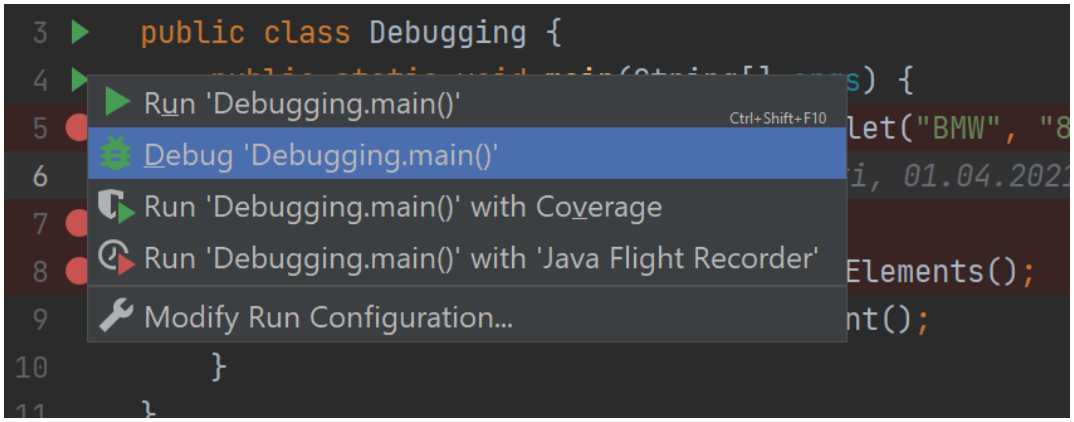
Żeby móc debugować program należy zacząć od breakpointu (te czerwone kropki).



Breakpoint wskazuje nam miejsce, w którym program ma się zatrzymać. Oczywiście potem możemy wznowić jego wykonywanie. Breakpoint możemy ustawić albo lewym przyciskiem myszki w danej linijce, musimy kliknąć bezpośrednio obok numeru linijki. Jak klikniemy na kodzie to nie postawimy breakpointu. Druga możliwość to mając kursor np. w linijce 4 należy wcisnąć `ctrl + F8`

Tryb debugowania

Mając już ustawione breakpointy, należy uruchomić program w trybie debugowania. Jeżeli uruchomimy go tak jak dotychczas, nie osiągniemy oczekiwanego efektu. W tym celu klikamy lewym przyciskiem myszki na strzałkę, którą normalnie uruchamiamy program i wybieramy opcję z robaczkiem.



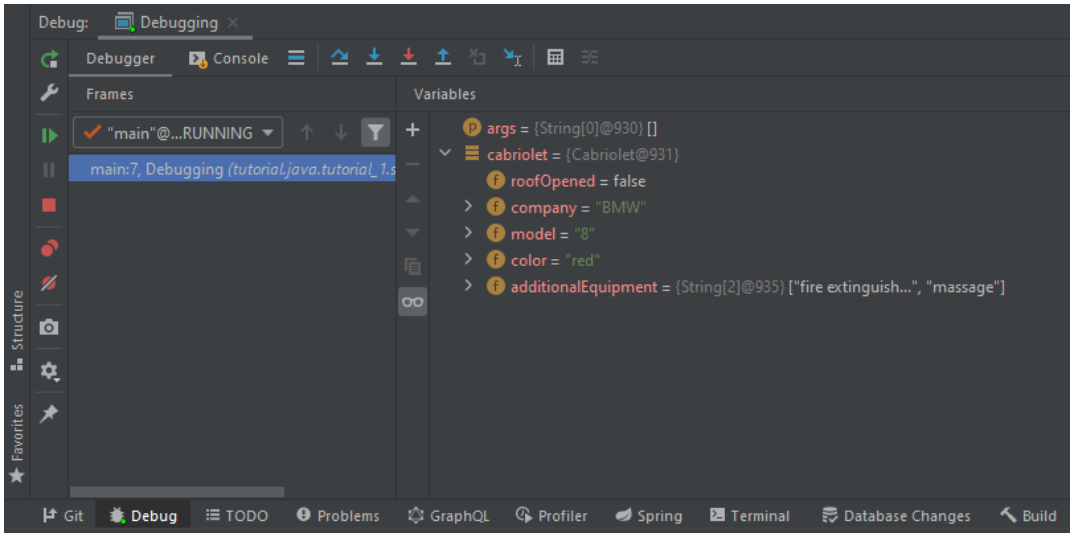
W tym momencie program uruchamia się w trybie debugowania. Możemy też uruchomić program w tym trybie, nie mając ustawionego żadnego breakpointu. Wtedy program wykona się normalnie, po prostu nigdzie się nie zatrzyma. Trzeba też pamiętać, że w trybie debugowania program może działać wolniej.

Jeżeli mieliśmy zapisaną już wcześniej konfigurację uruchomienia naszego programu, wystarczy, że wciśniemy robaczka w prawym górnym rogu ekranu. Albo skrót `shift + F9`.

Analizujemy stan wykonywania programu

Po uruchomieniu programu w trybie debugowania i zatrzymaniu wykonywania programu na jakimś breakpoincie pojawi nam się okno debuggowania.

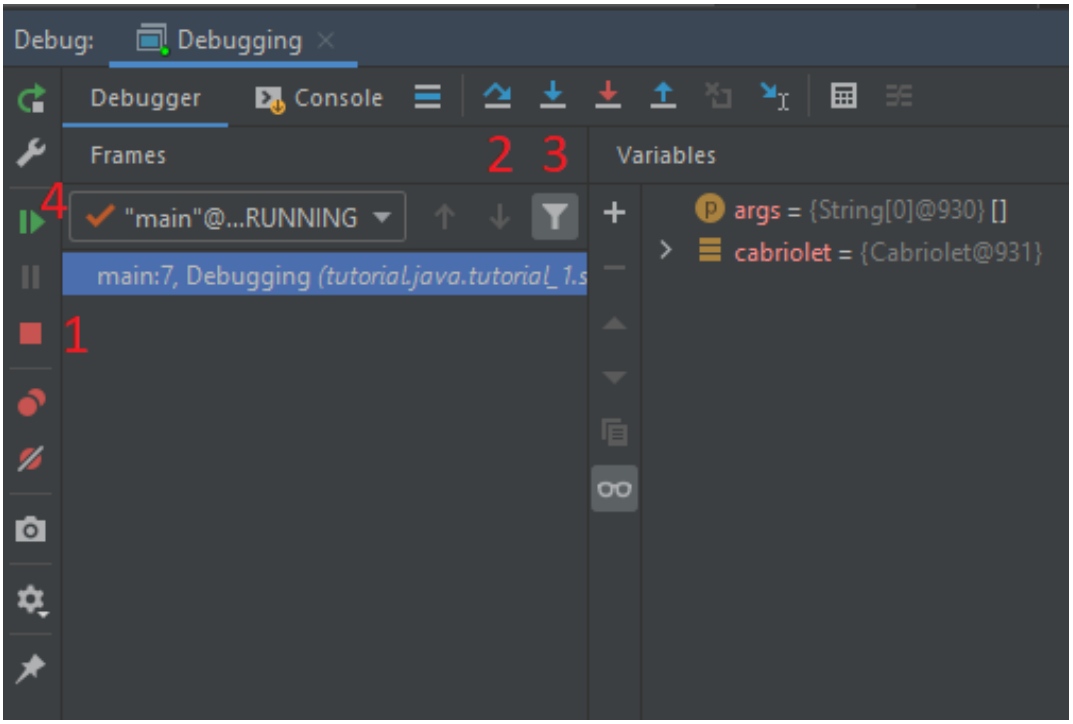
```
3   public class Debugging {
4       public static void main(String[] args) {   args: {}
5           Cabriolet cabriolet = new Cabriolet("BMW", "8", "red",
6               false, "fire extinguisher", "massage");
7           cabriolet.greetings();
8           cabriolet.closeRoof();   cabriolet: Cabriolet@931
9           cabriolet.sumAdditionalEquipmentElements();
10          cabriolet.printAdditionalEquipment();
11      }
12  }
```



Linijka w kodzie, która jest zakreślona na niebiesko (czyli nr 7) nie została jeszcze w naszym programie wykonana. Program będzie teraz wisiał w takim stanie do momentu, aż coś zrobimy. Czyli samo się teraz nie przestawi, trzeba coś kliknąć ☺. Możemy w tym momencie prześledzić wartości zmiennych, widzimy, że `company=BMW` lub `color=red`.

Przejście do kolejnej linijki

Jeżeli program znajduje się w stanie jak poprzednio, mamy możliwość kompletnie go zakończyć, bo znaleźliśmy już co chcieliśmy. W tym można wcisnąć czerwony kwadracik po lewej stronie ekranu (nr 1), program się wtedy zatrzyma.



Możemy też wykonać teraz program linijka po linijce przyciskając przycisk nr 2 (nad 2) lub skrót `F8`. Krok ten nazywa się `Step Over`. Program przejdzie teraz w swoim wykonywaniu do kolejnej linijki. Należy jednak pamiętać, że jeżeli po drodze naszego wykonywania pojawi się metoda, to robiąc `Step Over` pominiemy jej wykonywanie, nie wejdziemy 'do środka'. `Step Over` przechodzi przez następne kroki w kodzie, który widzimy, aż do zakończenia metody, w której jesteśmy.

Jeżeli chcielibyśmy wejść do takiej metody w trakcie wykonywania programu, możemy albo wybrać opcję `Step Into` (nr 3), albo skrót `F7`. Spowoduje to wejście wykonywania programu do metody, która będzie wykonana w następnym kroku. Drugi sposób na wejście do tej samej metody to postawić w niej breakpoint w interesującym nas miejscu i użycie `Resume Program` opisanego niżej.

W momencie, w którym teraz jesteśmy, możemy też wznowić wykonywanie programu, inaczej mówiąc, uruchomić go dalej albo do samego końca, albo do momentu napotkania kolejnego breakpointu. W tym celu należy wybrać opcję `Resume Program`, czyli przycisnąć przycisk nr 4 po lewej stronie ekranu (zielona strzałka) lub `F9`.

Przechodząc przez program w opisany sposób, dojdziemy do jego końca i możemy wtedy uruchomić debugger ponownie tak samo, jak był uruchomiony na samym początku.