Na początku stworzyłam sobie klasę Start, w której określiłam sobie, że tworzę sobie nową tablicę o określonej ilości elementów. Pętla for określa, że każda liczba (0,200> ma zostać wylosowana w randomowej kolejności i pokazana na ekranie razem z przecinkami. Następnie Start wyświetla menu, z którego mamy opcje wyboru, program skanuje nasz wybór z klawiatury i w zależności od wybranej opcji przeskakuje do wybranego sortowania. Menu zostało stworzone przy użyciu switch-case. Zostały określone tylko dwa rozwiązania, jeśli użytkownik wybierze coś innego program pokaże informacje o nieprawidłowym wyborze. W środku każdej opcji wyboru program ma napisane aby wyświetlić nazwę wybranego sortowania, następnie mówię mu, że to nowe sortowanie, czyli leci od początku ze wszystkimi 200 elementami mojej tablicy, program przesyła dane z mojej tablicy do klasy sortowania oraz zaczyna liczyć kroki.

Kolejny plik zawiera sortowanie bąbelkowe, które polega na porównywaniu ze sobą dwóch kolejnych elementów, następnie następuje zmiana tak aby mniejsza była ustawiona przed większą (np. 98765, program bierze 9 i 8 porównuje je, następuje zamiana ->89765 następnie bierze 9i7 porównuje je i przestawia itd. ) Tworzę int przechowywujące dane z liczenia kroków oraz określam, że każde wykonanie takiej zamiany będzie określone jako 1. w klacie public int[] zaczyna się sortowanie mojej tablicy. Podczas gdy dla każdej zamiany będziemy sumować zmianę jako 1, od tej chwili tablica będzie zawierała jedno miejsce mniej (n-1 -> n = wielkość tablicy). W pętli if zachodzi porównanie, o którym mowa była wcześniej i ustawienie liczby, sumujemy zamianę, sumujemy krok. Pętla określona w public int[] będzie się wykonywać na całej wielkości tablicy, to znaczy nie będzie to pojedyncze wykonanie komend, tylko na wszystkich 200 elementach. Po ustawieniu wszystkich elementów -> przechodzi z while do for -> program dla ustawionej tablicy wyświetla ją. Wyświetlony zostaje komunikat o zakończeniu sortowania. Program nie zwraca już żadnych wartości. Kolejna klasa public wyświetla naliczone kroki podczas sortowania i kończy działanie. Za każdym razem program przy wybieraniu losowania tworzy nową tablicę z nowymi liczbami w randomowych miejscach, aby uporządkować tablice program musi zrobić około 9tysięcy do 11 tysięcy kroków ( przykładowy wynik sortowania: 10715 ).

Następnym plikiem jest sortowanie kubełkowe, które polega na stworzeniu „kubełków”, każdego o innym zakresie liczbowym, następnie ustawieniu tych kubełków po kolei od najmniejszej wartości do największej, a następnie oddzielnego posegregowania zawartości wrzuconych do kubełka. Aby móc dokonać takiej segregacji na samym początku dodaję opcję Arrays, która pozwala mi na przechowywanie wielu takich samych danych ( jak int danych z których stworzona jest tablica) w jednej zmiennej. Następnie program otrzymuje informacje, że każdy dodany int będzie przechowywany w lenght, jest to nasz kubełek który będzie przechowywał dane ( np. jeśli wrzucimy dwie takie same liczby do kubełka o danym zakresie lenght wyniesie 2, jeśli do innego nie wrzucimy nic lenght wyniesie 0 itd.). W tym momencie tworzę też osobny int przechowywujący liczbę kroków. Komenda arrays.fill pozwala na uzupełnienie kubełka elementami z mojej tablicy o dane o określonym zakresiem, początkowo w każdym kubełku będzie zachowywane 0. Kolejna pętl for przechodzi przez wszystkie elementy tablicy, w tym momencie rozpoczyna się sortowanie. Program bierze więc liczbę i porównuje je z liczbami nieuporządkowanego kubełka -> arrays[step\_counting]. Jeśli wystąpi taka liczba w kubełku o indeksie, który jest równy tej liczbie podnoszę lenght o jeden. Kolejna pętla for wyświetla liczby w kubełkach, to znaczy biorę kolejne kubełki i sprawdzam jakie i ile liczb jest tam przechowywanych. Jeśli w kubełku 5 jest ukryta wartość 8 to znaczy, że musze 8 razy wyświetlić liczbę 5. Wyświetlam liczbę porównań wraz z posortowaną tablicą. Pętla for dla zawartości danego kubełka wyświetla liczbę kubełka odpowiednią ilość razy. Program ten wykonuje około 200 kroków sortowania.