Zad. 8: Fabryka obiektów i singleton

1 Cel ćwiczenia

Praktyczna realizacja wzorca projektowego *fabryki obiektów* i *singletona*. Utrwalenie umiejętności posługiwania się wskaźnikami dzielonymi i wykorzystanie techniki rzutowania *w górę* oraz polimorfizmu.

2 Program zajęć

- *Ocena realizacji zadania z poprzedniego laboratorium* – ocenie podlega poprawność realizacji zadania, styl pisania programu oraz dokumentacja wygenerowana za pomocą programu *doxygen*.
- *Ocena przygotowania do zajęć* – ocenie podlega diagram klas.
- *Modyfikacja programu wg wskazań osoby prowadzącej* – ocenie będzie podlegała poprawność realizacji modyfikacji. Pracę nad modyfikacją programu (wszystkie operacje należy wykonywać na kopii) należy rozpocząć już w trakcie pierwszej fazy laboratorium, gdyż prowadzący nie będzie w stanie ocenić wcześniejszego programu wszystkim jednocześnie.
- *Realizacja wstępnej fazy prac nad nowym zadaniem* – w ramach wstępnej realizacji zadania należy zaadoptować dostarczone przykłady prezentowane w ramach wykładu nr 11, do problemu opisanego w niniejszym zadaniu.
- *Ocena realizacji wstępnej fazy zadania*

3 Opis zadania programowego

Niniejsze zadanie jest dalszym rozszerzeniem wcześniejszego zadania. W programie powinna być możliwość dodawania obiektów takich jak przeszkody i roboty. W tym celu w kodzie programu należy stworzyć klasę, która będzie realizowała wzorzec projektowy *fabryki obiektów*. Obiekt pełniący rolę fabryki obiektów powinien być tylko jeden. Należy to zagwarantować w odpowiedniej konstrukcji tej klasy poprzez realizację wzorca projektowego *singleton*.

4 Przygotowanie do zajęć

Należy przygotować diagram klas proponowanych struktur danych oraz zapoznać się z przykładami dostarczonymi do wykładu nr 11.

5 Wymagania co do konstrukcji programu

Oprócz wymagań sformułowanych w opisie zadania należy uwzględnić uwarunkowania przedstawione poniżej.

- Należy zdefiniować klasę `FabrykaObiektow`, która implementuje wzorzec projektowy *fabryki obiektów* oraz *singletona*. Definicja klasy ma być zawarta w osobnym pliku nagłówkowym, zaś definicje metod w osobnym module.
- Wszystkie obiekty klasy `Robot`, `Sciezka` oraz `Przeszkoda` mają być tworzone w programie tylko i wyłącznie z wykorzystaniem singletona fabryki obiektów.
- Program powinien zawierać dodatkowe pozycje w menu, które pozwolą na dodanie robota lub przeszkody i określenie ich wstępnych parametrów. W przypadku robota będzie to położenie, zaś w przypadku przeszkody również jej rozmiary.

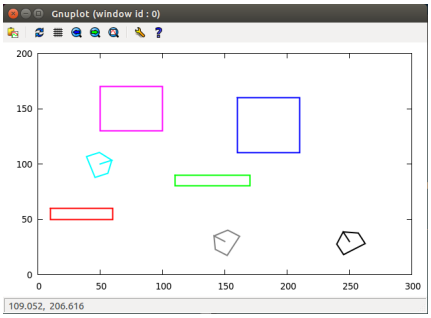
Program powinien zachować wszystkie funkcjonalności z poprzedniej wersji. Należy więc pamiętać, że:

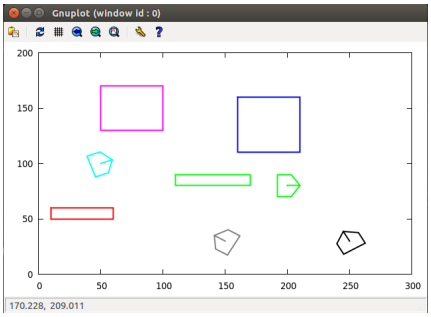
- Tak jak we wcześniejszej wersji programu wszystkie obiekty klasy `Robot`, `Sciezka` oraz `Przeszkoda` należy umieścić na jednej liście.
- Jako wskaźników w liście liście należy użyć wskaźników *inteligentnych* (`shared_ptr<>`).
- W obiekcie klasy `ObiektGraficzny`, podobnie jak w obiekcie klasy `Wektor2D` należy wprowadzić pole statyczne, które pozwoli zliczyć ile łącznie zostało utworzonych tego typu obiektów, oraz ile ich pozostało na końcu działania programu. W dobrze skonstruowanym programie, gdy program kończy swoje działanie, nie powinno już być żadnego obiektu tego typu.

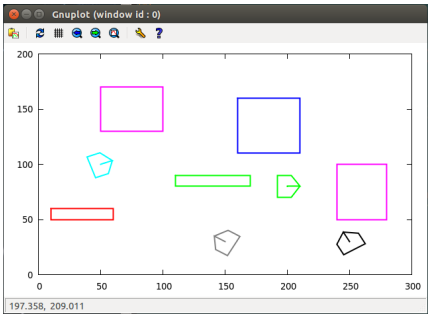
Oprócz tego pozostają w mocy wszystkie wcześniejsze wymagania dotyczące struktury katalogów, pliku Makefile, modułowej struktury programu, jak też opisów i generacji dokumentacji za pomocą systemu `doxygen`.

6 Przykład działania programu

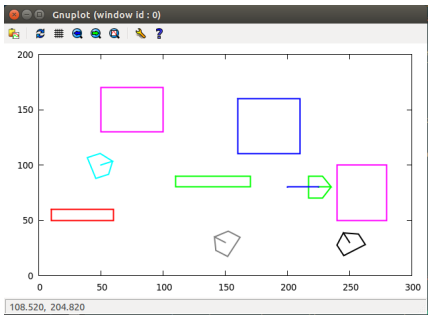
```
jkowalsk@panamint> ./roboty_przeszkody
 Laczna ilosc stworzonych obiektow klasy Wektor2D: 1568
       Ilosc istniejacych obiektow klasy Wektor2D: 124

Aktualnie wyselekcjonowanym robotem jest:
  Robot 2.   Wspolrzedne: (250, 30)

  R - dodaj robota
  P - dodaj przeszkode
  z - zmiana szybkosci ruchu robota
  o - obrot robota
  j - jazda na wprost
  s - selekcja robota

  t - zadaj translacje rysunku
  p - powrot do pierwotnego ustawienia rysunku

  w - wyswietl ponownie menu

  k - zakoncz dzialanie programu

 Twoj wybor (w - wyswietl menu)> R

  Podaj docelowe wspolrzedne robota: x y > 200 80

 Twoj wybor (w - wyswietl menu)> P

  Podaj docelowe wspolrzedne przeszkody (lewy gorny rog)
  oraz jej szerokosc i dlugosc: x y sz wys > 240 100 40 50

 Twoj wybor (w - wyswietl menu)> s

Aktualnie wyselekcjonowanym robotem jest:
  Robot 2.   Wspolrzedne: (250, 30)


        0 - zaniechaj zmiany selekcji

  Robot 1.   Wspolrzedne: (50, 100)
  Robot 2.   Wspolrzedne: (250, 30)
  Robot 3.   Wspolrzedne: (150, 30)
  Robot 4.   Wspolrzedne: (200, 80)

 Podaj numer robota, dla ktorego maja być wykonane operacje sterowania

 Wprowadz numer robota lub 0 > 4

  Robot 4.   Wspolrzedne: (200, 80)
```







```
 Twoj wybor (w - wyswietl menu)> j

Aktualnie wyselekcjonowanym robotem jest:
  Robot 4.   Wspolrzedne: (200, 80)

 Podaj dlugosc drogi ruchu robota na wprost.
 Dlugosc drogi: 100

 !!! Ruch nie moze być kontynuowany ze wzgledu
 !!! na wystapienie kolizji.
```




```
 Twoj wybor (w - wyswietl menu)> k

  Laczna ilosc stworzonych obiektow klasy ObiektGraficzny: 13
       Ilosc nieusunietych obiektow klasy ObiektGraficzny: 0

jkowalsk@panamint> _
```

7 Materiały pomocnicze

Do tego zadania nie jest dostarczany żaden zalążek. Pracę należy rozpocząć wykorzystując program z wcześniejszego zadania nr 7.

W na `panamincie` w podkartotece `~bk/edu/kpo/zad/z8` można znaleźć skrypt, który uruchamia program demonstrujący przykładową realizację zadania. Skrypt ten można uruchomić poleceniem:

```
~bk/edu/kpo/zad/z8/przyklad_rozwiazania.sh
```

Ze względu na to, że program tworzy okienko graficzne, w przypadku połączenia z panamintem za pomocą programu `ssh` należy pamiętać, aby użyć opcji `-X`.

8 Rozszerzenia

Możliwych jest kilka wariantów rozszerzenia tego zadania.

8.1 Możliwość usuwania przeszkód w trakcie działania programu

Program powinien umożliwiać usuwanie wskazanej przeszkody lub robota.

8.2 Graficzna selekcja elementów

Operacja selekcji robota powinna powodować zmianę sposobu rysowania robota, np. zwiększenie grubości linii.

8.3 Szablon wektora

To rozszerzenie będzie najwyżej punktowane. Należy zdefiniować własny szablon wektora i używać go w programie zamiast wcześniejszej klasy `Wektor2D`.

9 Wymagania i zarys programu zajęć w okresie realizacji zadania

Przystępując do pracy nad programem zaleca się, aby rozszerzenie menu programu zrealizować na samym końcu, gdy stworzymy już i przetestujemy wszystkie niezbędne funkcjonalności.

9.1 Tydzień 0

Należy zaadaptować dostarczone przykłady prezentowane w ramach wykładu nr 11, do problemu opisanego w niniejszym zadaniu.

9.2 Tydzień 1

Przed zajęciami muszą zostać przygotowane następujące elementy zadania. Wszystko co będzie ponad to będzie oceniane *in plus* (oprócz menu programu).

- Zdefiniowane powinna być klasa `FabrykaObiektow`. Dla tej klasy powinien istnieć osobny moduł. Należy zdefiniować też metodę do tworzenia osobnych obiektów klas `Robot`, `Sciezka` oraz `Przeszkoda`.
- Klasa `FabrykaObiektow` powinna być użyta w poprzedniej wersji programu. Powinien on zostać doprowadzony do stanu, aby działał co najmniej tak jak program ze wcześniejszego zadania, tzn. aby były dostępne wszystkie funkcjonalności.
- Warunkiem koniecznym pozytywnej oceny jest poprawna kompilacja. W trakcie kompilacji nie powinny być generowane żadne ostrzeżenia.
- Nowa klasa i jej metody muszą być opisane oraz powinna być wygenerowana dokumentacja za pomocą programu `doxygen`.

9.3 Tydzień 2

- Rozliczenie się z gotowego programu.