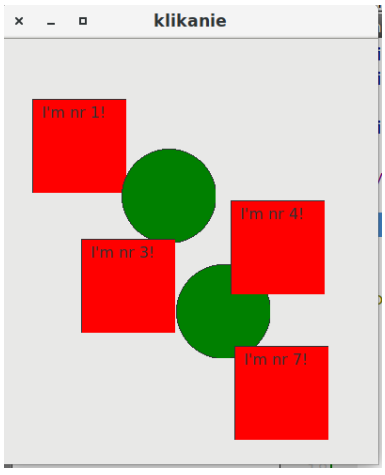
Lista 6 – funkcje wirtualne

1. (4 pkt) Skompiluj i uruchom załączony program (korzystający z biblioteki Qt) składający się z plików:
   - `main.cpp`
   - `mainwindow.h`
   - `mywidget.h`
   - `mainwindow.cpp`
   - `mywidget.cpp`
   - `zad1.pro`

   Programy napisane w Qt najprościej kompilować w programie QtCreator, jak na wykładzie (w QtCreatorze otwiera się plik `*.pro`, a potem trzeba się domyślić lub skonsultować z kolegą/koleżanką/stackoverflow/dokumentacją programu, co dalej).
   Następnie rozszerz ten program o nastepujące funkcjonlności:

- (1 pkt) Program po kliknięciu prawym klawiszem powinien umieszczać w obszarze roboczym okna inny obiekt niż czerwony kwadrat, np. zielony trójkąt, prostokąt, koło, wycinek koła etc. tak, by działanie lewego klawisza było inne niż prawego. W tym celu dodaj do programu nową klasę wzorowaną na `MyWidget`, ale reprezentującą inne (nowe) obiekty.
- (1 pkt) Program po przyciśnięciu `ctrl-z` powinien usuwać ostatnio dodany obiekt, z możliwością cofnięcia się do pierwszego obiektu i usunięcia go („undo”). W tym celu w klasie MainWindow możesz przeciążyć odpowiednią **funkcję wirtualną**. Jaką? Poszukaj w dokumentacji.
- (1/2 pkt) Jeżeli użytkownik przyciśnie `ctrl-z` w sytuacji, gdy liczba wyświetlanych widżetów równa jest zero, program powinien wyświetliś jakiś komunikat ostrzegawczy (proste okienko dialogowe, por. `main.cpp`) i oczywiście niczego nie usuwać.
- (1/2 pkt) Upewnij się, że nie masz wycieków pamięci. W tym celu do klasy widżetów dodaj destruktory wyświetlające komunikaty diagnostyczne. Niestety, w Qt nie uzywa się `std::cout`. Zamiast tego użyj `qDebug`, np. tak:
  ```
  qDebug() << "destructor of " << this;
  ```
  Nie zapomnij nawiasów okrągłych po `qDebug`! Dla nabrania większej wprawy, analogiczną diagnostykę umieść też w konstruktorach.
- (1 pkt) Napisy na czerwonych kwadratach są takie nudne! Zmień implementację tak, by każdy czerwony kwadrat generowanego spod lewego klawisza myszki miał swój unikatowy numer wyświetlany na kawdracie, np. pierwszy kwadrat mógłby wyświetlać „`I'm nr 1`” etc.
  To, jak kontrolować numerację widżetów, musisz wymyślić sam(a).
  Do zamiany liczby na napis można użyć `std::to_string`, ale ja polecam `QString` i jej składową `arg`. Opis jej użycia: dokumentacja Qt.



Generalnie to zadanie jest trudne w tym sensie, że nikt z nas nie zna nawet 10% Qt. Z drugiej strony programowanie w dużym stopniu polega na umiejętności korzystania z istniejących narzędzi (np. QtCreator), bibliotek (np. Qt) i ich dokumentacji. W tym sensie zadanie jast BARDZO życiowe, tym bardziej że QtCreator i Qt są naprawdę popularne. Wszystko, czego potrzebujecie, znajduje się w dokumentacji Qt. Powodzenia!

2. (3 pkt.) W dokumentacji klasy `QObject` znajdujemy iformację, że w klasie tej dostępna jest metoda **`int QObject::`**`startTimer`. Wykorzysytaj ją do wprowadzenia do programu z poprzedniego punktu jakiejś prostej animacji, oczywiście za pomocą odpowiedniej **funkcji wirtualnej**.

Zadania dodatkowe (niepunktowane)

1. Uruchom swój program w debugerze. Ustaw pułapki w swoich implementacjach funkcji wirtualnych. Sprawdź, że są one wywoływane bezpośrednio przez Qt a nie przez Twoje funkcje.
2. W C++11 wprowadzono słowo kluczowe override. Jaką ono pełni funkcję? Gdzie zostało użyte w moim programie i czy użył(a/e) go także w swoim rozwiązaniu?
3. Sprawdź, co się stanie, gdy w `main.cpp` usuniesz instrukcję `return a.exec();`
4. Powitalne okienko dialogowe wyświetlane jest w `main.cpp` instrukcją QMessageBox::information. Dlaczego w zapisie tej funkcji użyto operatora :: ? Sprawdź w dokumentacji Qt, jakie atrybuty ma funkcja QMessageBox::information i jak wpływa to na sposób jej wywoływania.

```
int QMessageBox::information(QWidget *parent, const QString &title, const QString &text, int button0, int button1 = 0, int button2 = 0)  [static]
```

5. Gdybyś był deweloperem funkcji QMessageBox::information, która ma atrybut `static`, czy mógłbyś/mogłabyś użyć w jej implementacji słowa kluczowego **`this`**?
6. Ostatnie dwa argumenty funkcji QMessageBox::information oprócz typu (`int`) mają także wartość (`= 0`). Co oznacza ten zapis?
7. Gdzie na dysku znajduje się program wykonywalny do twojego rozwiązania? Uruchom swój program z konsoli.
8. Co się stanie, gdy z `main.cpp` usuniesz instrukcję `w.show()`? Czy po kliknięciu OK w oknie powitalnym program będzie działał w tle, czy też zostanie zakończony? Jak to sprawdzić?