BUILDER I PROTOTYP – termin wykonania zadania 19 listopada

a) Wyobraźmy sobie aplikacje typu USOS, do której mogą logować się zarówno studenci jak i administratorzy. Lista uprawnień/ przycisków jest inna w przypadku administratorów a inna dla studentów jak również inne jest tzw. 'Welcome Message'.
Stwórz następującą hierarchie klas:
Klasa FormBuilder z metoda constructForm (uzywajaca do stworzenia formatki ponizszych klas/metod:
Klasa abstrakcyjna ButtonsBuilder definiująca interfejs addButtons().
Jej klasy pochodne StudentsButtonsBuilder, AdminButtonsBuilder
Klasa abstrakcyjna WelcomeMessageBuilder definiująca interfejs printWM().
Jej klasy pochodne StudentsWMBuilder, AdminWMBuilder
Potrzebna będzie też klasa Form reprezentująca formatkę aplikacji zawierająca pole odpowiadające WelcomeMessage oraz jakąś strukturę danych przechowującą listę dostępnych przycisków.

```
class BuilderExample {
  public static void main(String[] args) {
    FormBuilder fBuilder = new FormBuilder ();
    StudentsButtonsBuilder bBuilder = new StudentsButtonsBuilder();
    StudentsWMBuilder wmBuilder= new StudentsWMBuilder ();

    fBuilder.setButtonsBuilder(bBuilder);
    fBuilder.setWMBuilder(wmBuilder);

    fBuilder.constructForm();

    Form form = fBuilder.getForm();
  }
}
```

b)Utwórz abstrakcyjna klasę Multimedia z deklaracjami metod clone() oraz toString(); jeżeli chodzi o pola to powinna zawierać przynajmniej nazwę (np domyslną) jak i typ obiektu. Proszę dodać przynajmniej jedno pole wskaźnikowe (C++) lub jego odpowiednik tak, żeby konieczne było stworzenie deep copy obiektu przy klonowaniu.
Następnie utwórz klasy do niej pochodne Picture, Music, Movie.
Klasy powinny zadziałać z poniższym kodem (lub analogicznym w Java)

```
int main(){
        std::vector<Multimedia*> my_multimedia;
        my_multimedia.push_back(new Picture);
        my_multimedia.push_back(new Music);
        my_multimedia.push_back(new Movie);

        std::vector<Multimedia*> multimedia_copy;

      for(std::vector<Multimedia*>::iterator it = my_multimedia.begin();
          it != my_multimedia.end();
          ++it)
      {
                multimedia_copy.push_back( (*it)->clone() );
      }
      for(std::vector<Multimedia*>::iterator it2 = my_multimedia.begin();
          it2 != my_multimedia.end();
          ++it2)
      {
                (*it2)->toString();
      }
      return 0;
}
```