ZPO 02. (Zasady SOLID)

Foldery “solid _1” and “solid_ 2” zawierają dwa przykłady kodu, napisanego dla następującej sytuacji:

> *The inhabitants of the Green Dragon city can play various roles, e.g. the role of a gardener, a knight etc. A gardener plants trees when the weather is fine and looks at the growing trees when it is raining. A knight fights with a weapon (e.g. sword, bow) and rides on a mount (e.g. horse, dragon) and his strength decreases during the fight, so he must take a rest when he becomes tired.*
>
> *Each inhabitant is characterized by his/her name, gender, and the names of his/her parents. An inhabitant (e.g. Mr Greenleaf, son of Brightstar and Bravewolf) can play various roles, for example, he can be a knight, and then change to a wizard.*
>
> *The Master of the city stores the information about all the inhabitants (the list of all the inhabitants), can show all the inhabitants (method* `show()`*) and can make them play their roles (method* `play()`*). He can also change the roles of the inhabitants.*

Zadania

1. Narysuj diagramy klas UML dla programów “solid_1” and “solid_2”.
2. Wskaż i wyjaśnij (nie pisz kodu), jakie zmiany należałoby wprowadzić w kodzie programów “solid _1” i “solid_2” w następujących przypadkach (tzn. co trzeba dodać/zmodyfikować w klasach, metodach itp.):
   a) kiedy do cech mieszkańca dodany zostanie nowy atrybut `date_of_birth`;
   b) kiedy utworzona zostanie nowa rola, np. czarodzieja, który posiada metodę `cast_a_spell`;
   c) kiedy zarządca miasta będzie chciał przydzielić nową rolę do osoby, która obecnie “pracuje” jako rycerz.
3. Czy programy “solid_1” i “solid_2” spełniają zasady SOLID? Jeżeli nie, to wskaż, które z zasad nie są spełnione i wyjaśnij dlaczego.
4. Zaproponuj projekt programu, który umożliwi łatwą (zgodną z zasadami SOLID) zmianę ról mieszkańców (osób), np. osoba o imieniu Quickhand będąca obecnie rycerzem będzie mogła „awansować” na stanowisko czarodzieja. Narysuj odpowiedni diagram klas UML i napisz kod.