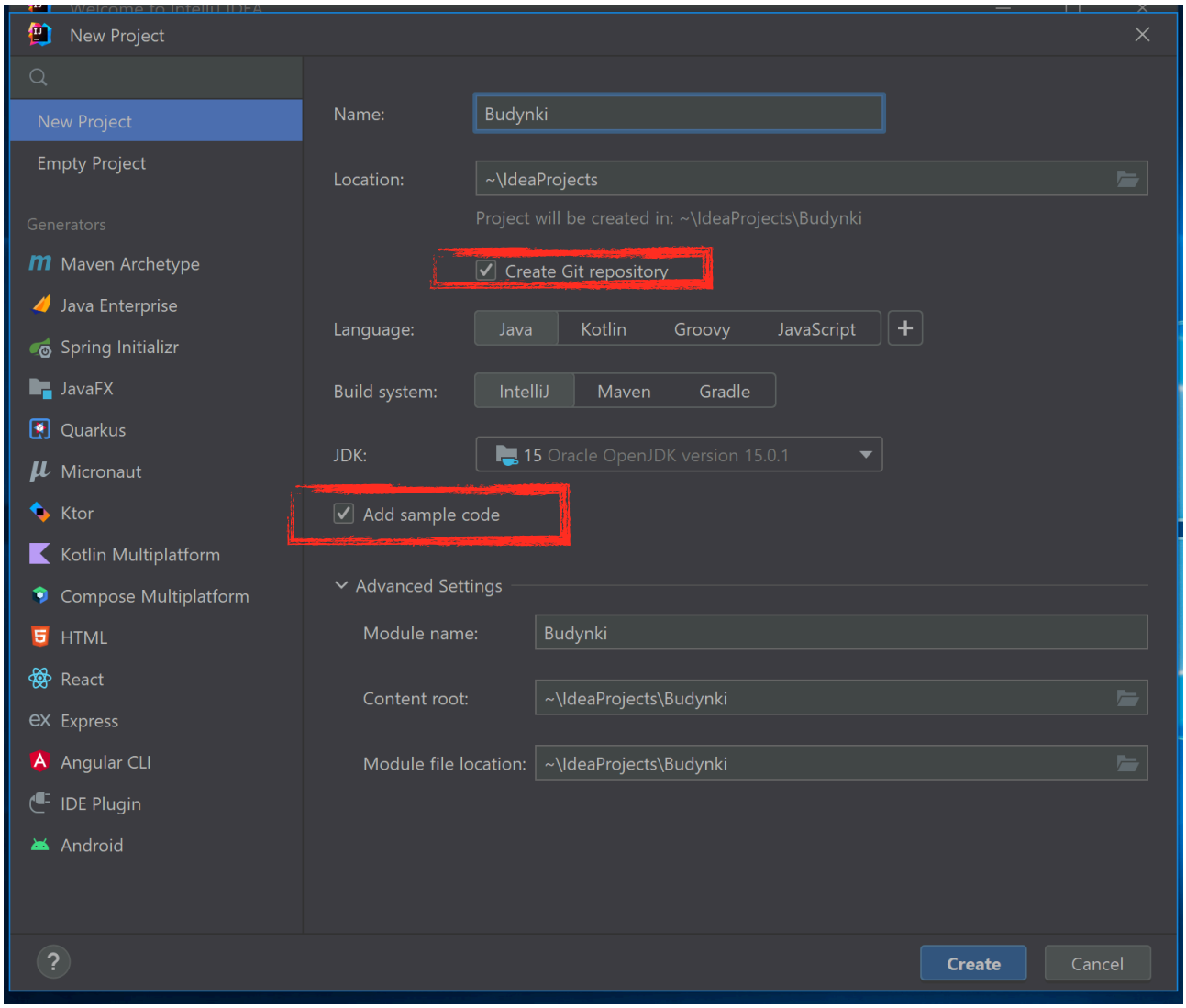
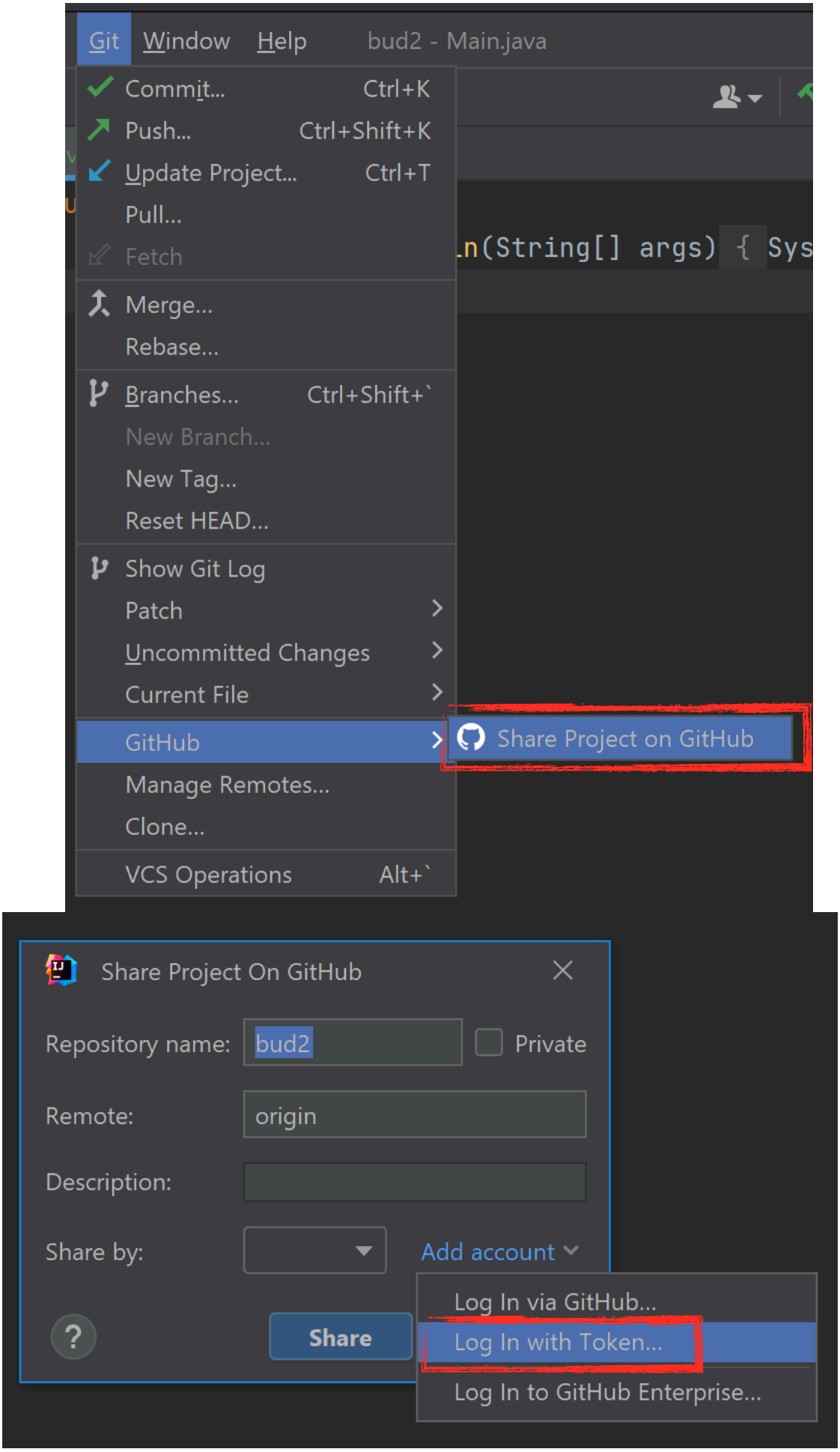
Zastosowania informatyki w pracy grupowej

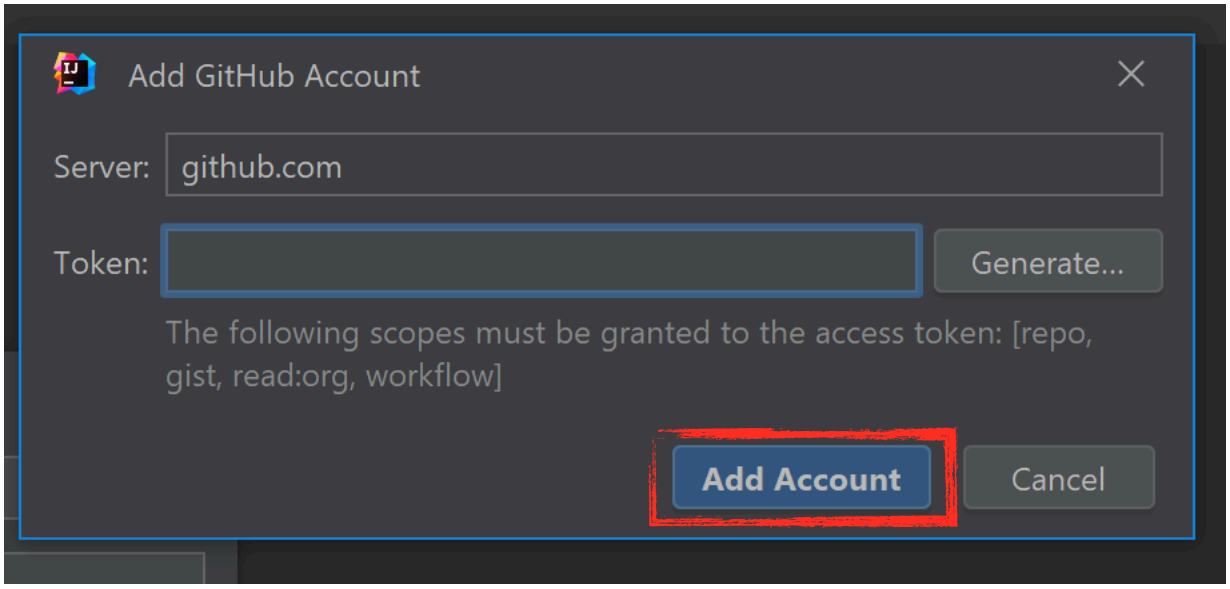
Laboratorium 3 - GitHub IntelliJ

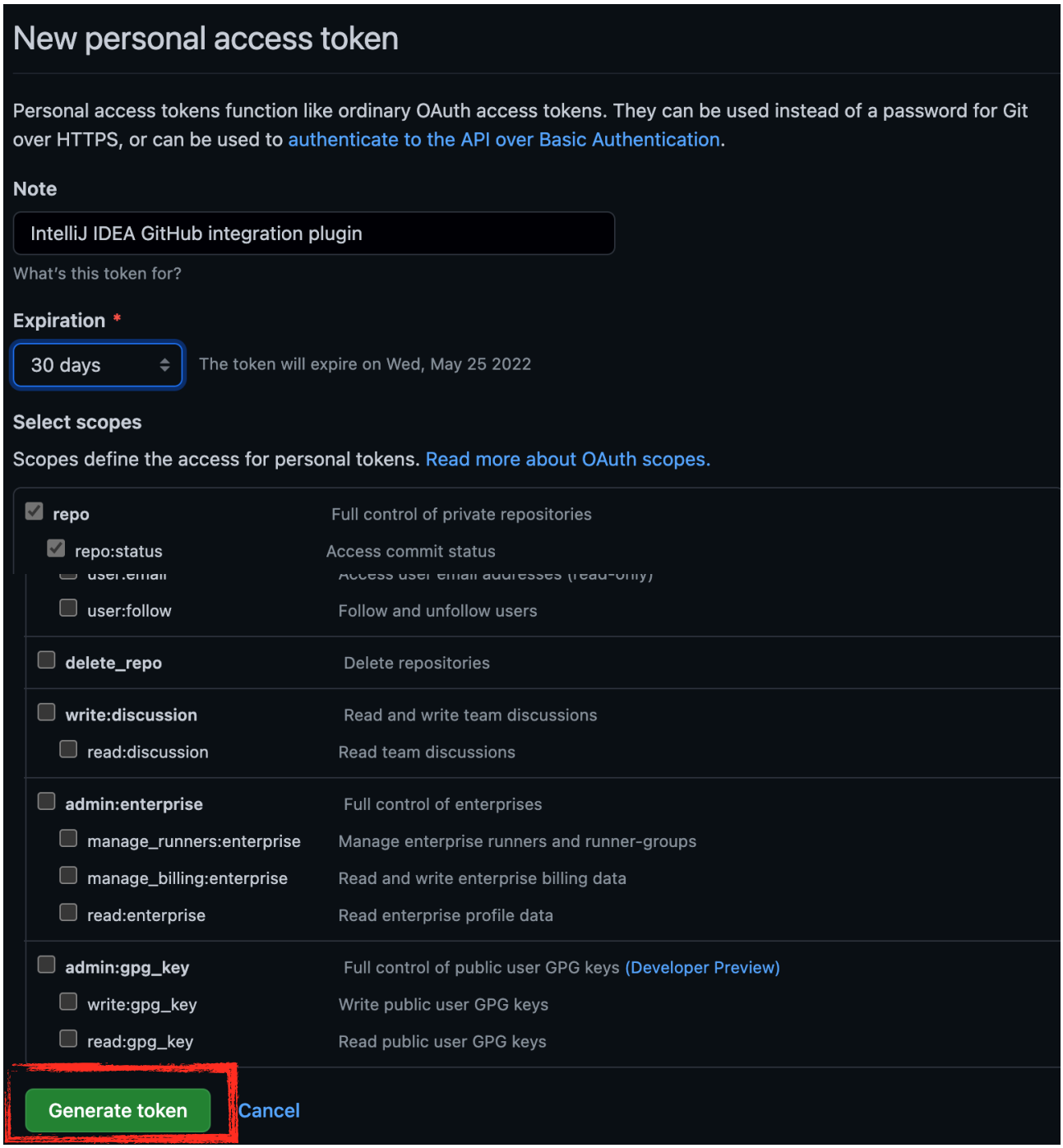
1. Proszę dobrać się w 3 osobowe grupy.
2. Wspólnie za pomocą IntelliJ oraz GitHuba opracować program w języku Java. Ma być to aplikacja konsolowa demonstrująca użycie klas w programowaniu obiektowym.
3. Członkowie grupy muszą stworzyć konta na GitHubie.
4. Jedna z osób tworzy nowy projekt w IntelliJ



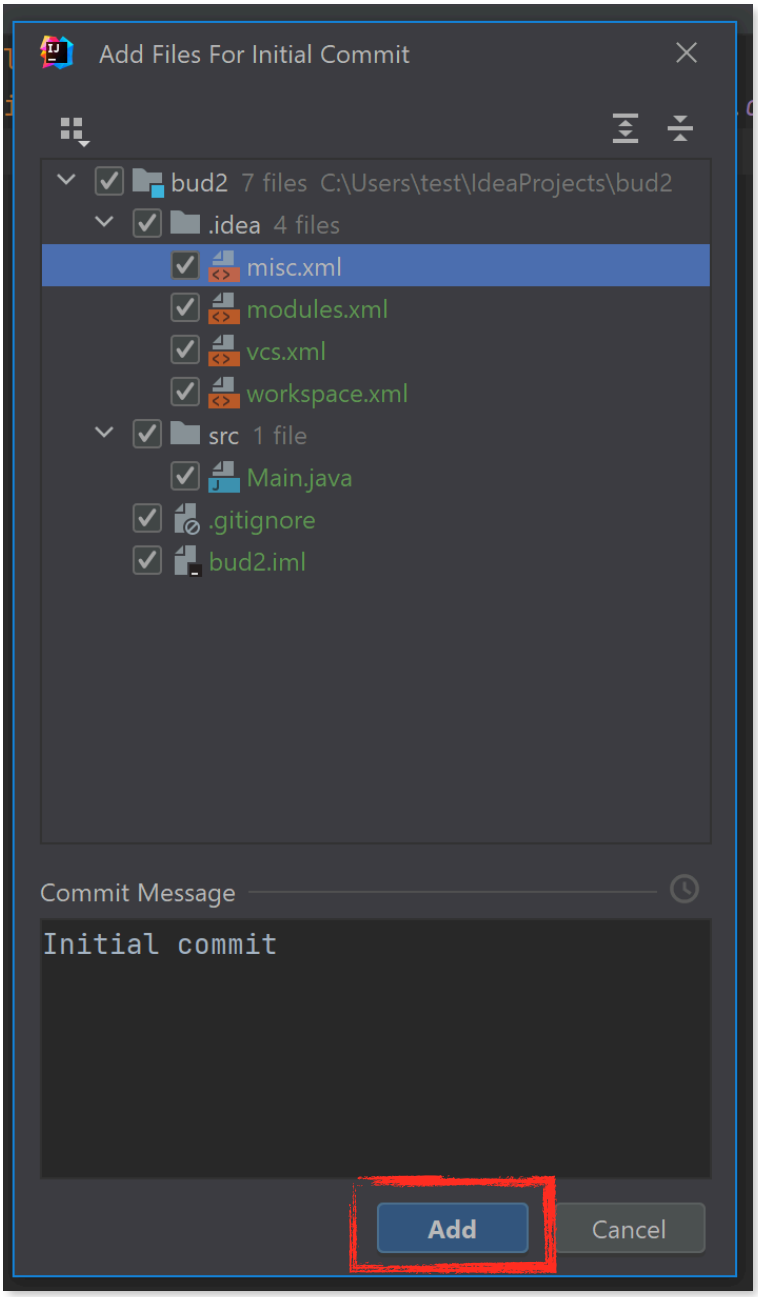
5. Następnym krokiem powinno być podłączenie konta GitHub do IntelliJ




Add GitHub Account
Server:
github.com
Token:
Generate...
The following scopes must be granted to the access token: [repo, gist, read:org, workflow]
Add Account
Cancel

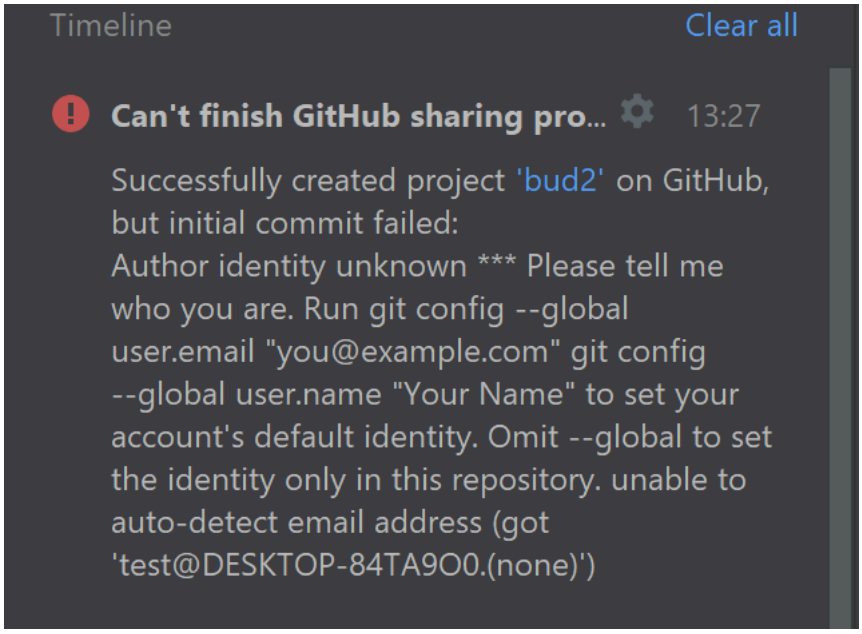

New personal access token
Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.
Note
IntelliJ IDEA GitHub integration plugin
What's this token for?
Expiration *
30 days
The token will expire on Wed, May 25 2022
Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo
Full control of private repositories
repo:status
Access commit status
user:follow
Follow and unfollow users
delete_repo
Delete repositories
write:discussion
Read and write team discussions
read:discussion
Read team discussions
admin:enterprise
Full control of enterprises
manage_runners:enterprise
Manage enterprise runners and runner-groups
manage_billing:enterprise
Read and write enterprise billing data
read:enterprise
Read enterprise profile data
admin:gpg_key
Full control of public user GPG keys (Developer Preview)
write:gpg_key
Write public user GPG keys
read:gpg_key
Read public user GPG keys
Generate token
Cancel

6. Następnym krokiem jest wykonanie pierwszego commita



Wybranie folderu .idea do commita spowoduje zmieszczenie w repozytorium także całej konfiguracji projektu dla IntelliJ, z której bedą mogli skorzystać współpracownicy. Zależnie od zaawansowania programisty może to być dobre lub złe rozwiązanie. Zazwyczaj w poważnych projektach takiej konfiguracji się nie zawiera w repozytorium.

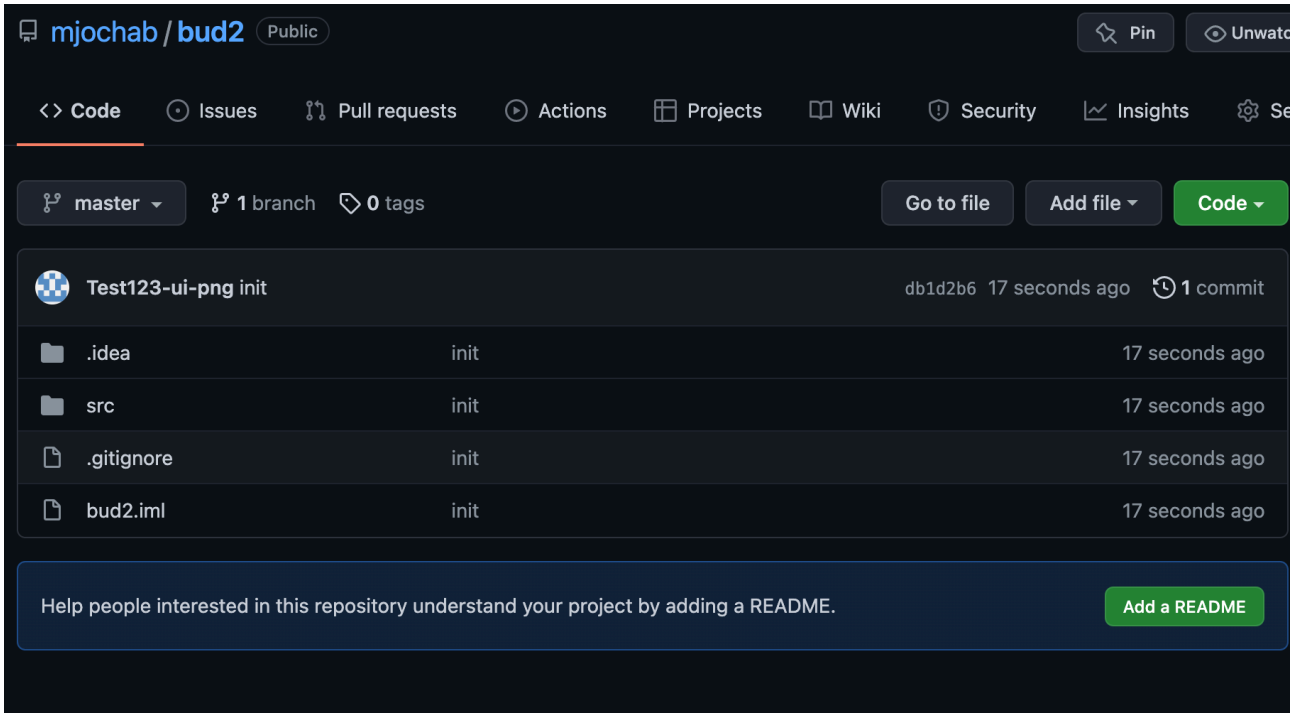
7. Jeżeli Git nie był wcześnie skonfigurowany możemy dostać poniższy komunikat:



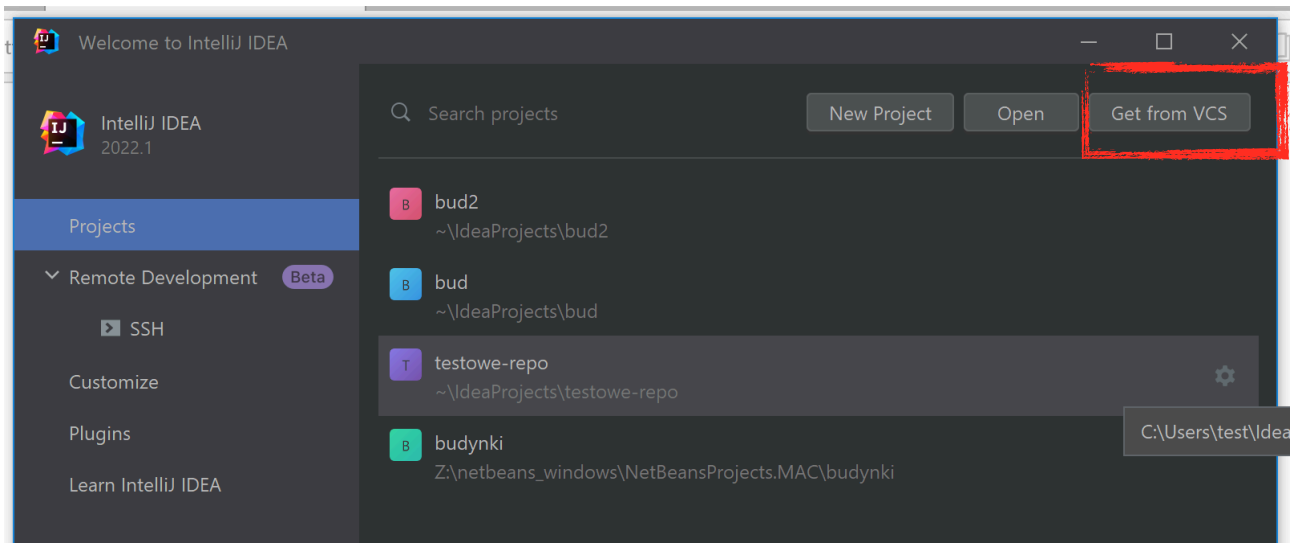
Należy wtedy:

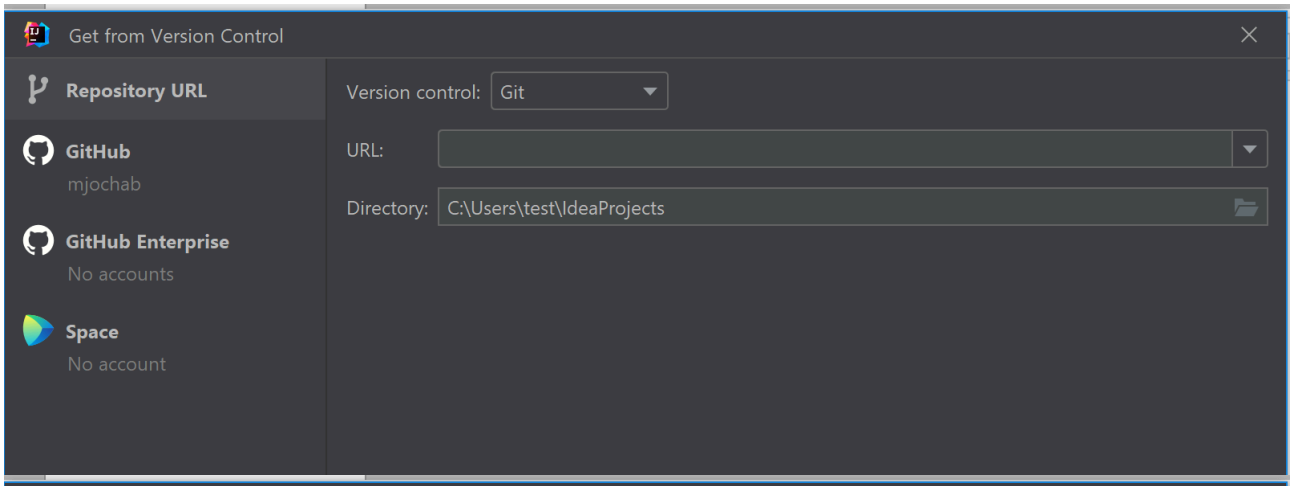
1. Wejść w Git/commit
2. Uzupełnić nazwę commita inicjalizującego
3. Kliknąć „Commit and Push…”
4. Uzupełnić brakującą nazwę i maila

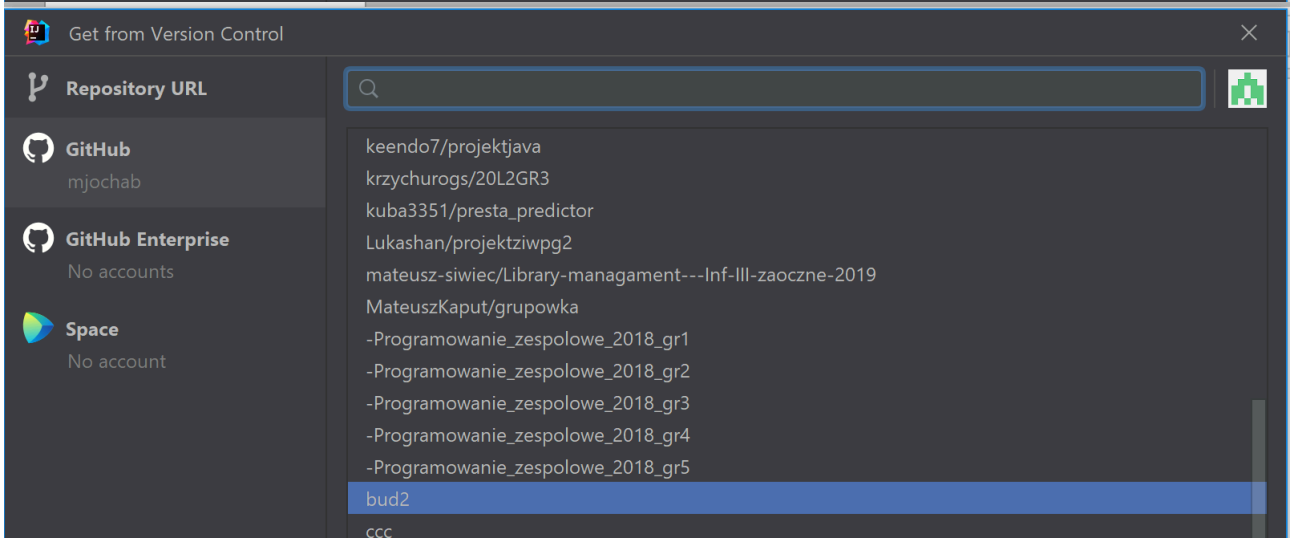
8. Jeżeli wszystko przeszło prawidłowo na stronie GitHub powinnismy zobaczyć pliki projektu:



9. Na GH należy wejść w *Settings/Collaborators* i dodać pozostałe osoby z grupy.
10. Wszyscy użytkownicy mogę teraz rozpocząć pracę nad projektem klonując projekt na swoje komputery korzystając z integracji z HitHub lub podając url repozytorium:







11. Proszę podzielić się pracą. Należy opracować:
    1. plik *README.md* ze skrótową dokumentacją - głownie instrukcją użycia aplikacji (w języku Markdown),
    2. Program ma składać się z następujących klas:
        1) interfejsu Powierzchnia zawierającego metodę policzPole, która będzie liczyć pole danego obiektu (uwzględniając podobiekty)
        2) Budynek:
            a) implementując interface Powierzchnia
            b) zawierając pola z:
                i. wymiarem działki
                ii. ilością pięter
                iii. listą pięter typu Pietro
                iv. polem typu Garaz

3) Pietro:
   a) implementując interface Powierzchnia
   b) zawierając pola z:
      i. ilością pomieszczeń
      ii. numerem piętra
      iii. tablicą ElementPietra
4) ElementPietra:
   a) abstrakcyjna
   b) implementując interface Powierzchnia
   c) zawierająca pole z numerem piętra
5) Mieszkanie:
   a) dziedzicząca po ElementPietra
   b) zawierając pola z:
      i. wizytówką z imieniem i nazwiskiem mieszkańca
      ii. tablicą pomieszczeń
6) Pomieszczenia:
   a) dziedzicząca po ElementPietra
   b) zawierając pola z:
      i. wymiarem x
      ii. wymiarem y
      iii. nazwą
7) Garaz:
   a) dziedzicząca po Pomieszczenie
   b) zawierając pola z:
      i. wymiarem x bramy
      ii. wymiarem y bramy

12. Dla każdego zadania należy utworzyć osobną gałąź poprzez *Git/ New Branch*. Dla innych osób będzie ona dostępna dopiero po *commit* i *push*.

Proszę pamiętać o sensownym nazewnictwie commitów i gałęzi!!!!

13. Aby pracować na gałęzi stworzonej przez kogoś innego należy najpierw wybrać Git*/Pull* i wybrać interesujące nas gałęzie celem

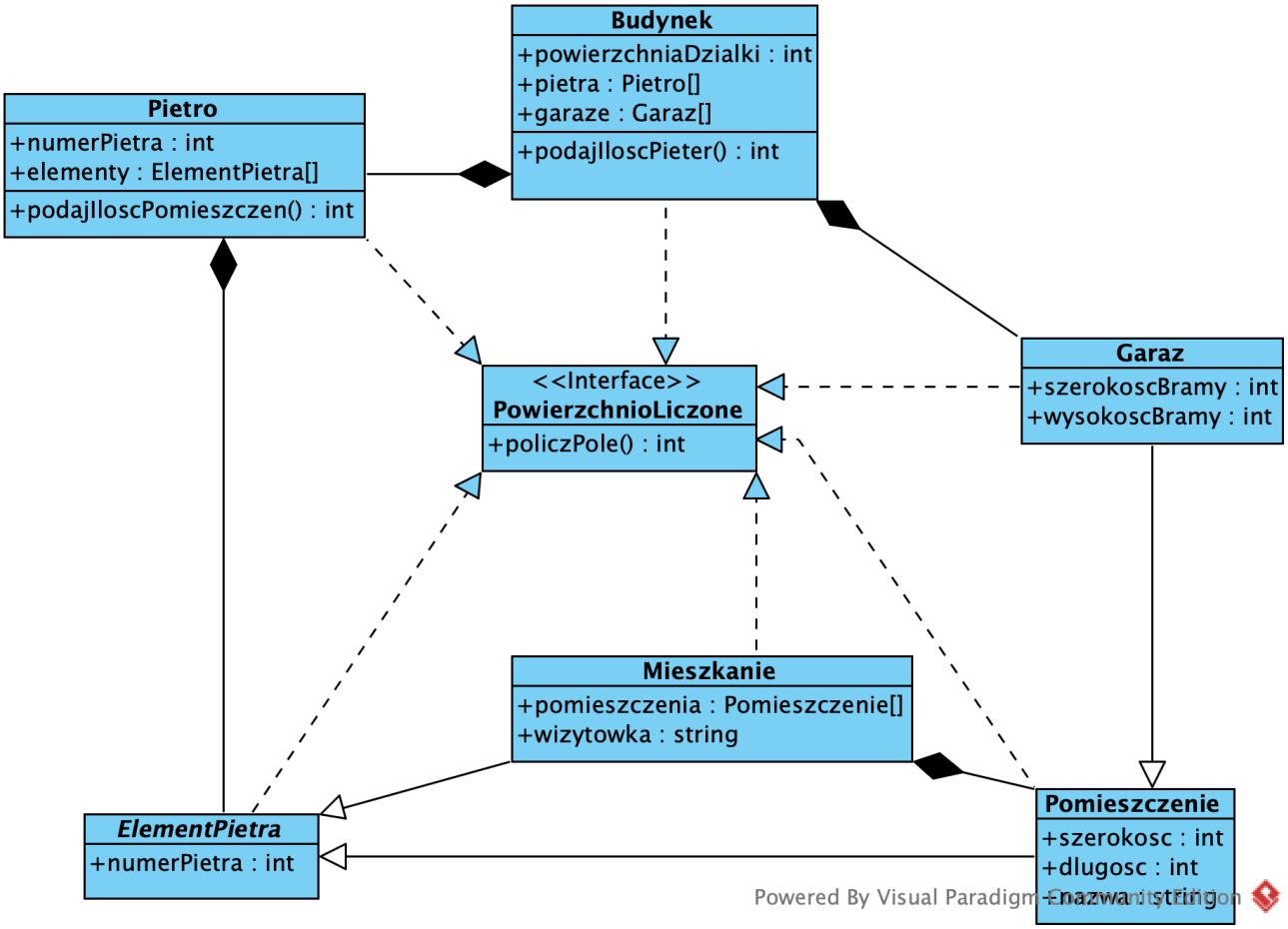
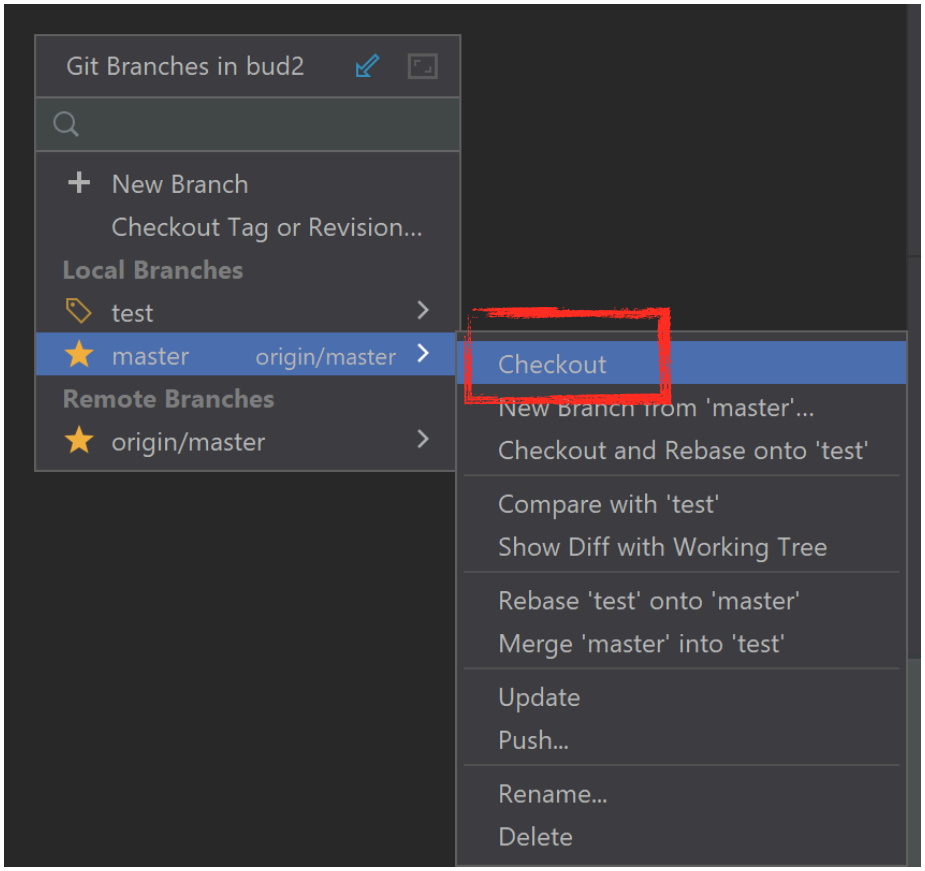


DIAGRAM UML KLAS

synchronizacji po czym przełączenia dokonuje się poprzez Git/ *Branch :*



14. W odpowiednim momencie dla każdego zadania proszę poprzez stronę GH stworzyć *Pull Request* i otworzyć dyskusję.
15. Po akceptacji przez innych należy dokonać *merge*. Wskazane jest poeksperymentowanie z różnymi ich rodzajami.
16. Zaobserwowane problemy należy zgłaszać poprzez *Issues*.
17. Wersje będące pewnym całościowym etapem można zaznaczać tagami.
18. Prawidłowa implementacja kodu powinna pozwolić na działanie kodu jak poniżej:

```
public static void main(String[] args) {

    Pomieszczenie[] mojePomieszczenia = {
        new Pomieszczenie() {
            {
                this.szerokosc = 3;
                this.dlugosc = 2;
                this.nazwa = "pokoj";
            }
        },
        new Pomieszczenie() {
            {
                szerokosc = 2;
                dlugosc = 2;
                nazwa = "lazienka";
            }
        }
    };

    Mieszkanie mojeMieszkanie = new Mieszkanie() {
        {
    ElementPietra[] mojeElementyParteru = {
        mojeMieszkanie,
        new Pomieszczenie() {
            {
                szerokosc = 2;
                dlugosc = 2;
                nazwa = "suszarnia";
            }
        }
    };
```

```
    Pietro[] mojePietra = {
        new Pietro() {
            {
                numerPietra = 0;
                elementy = mojeElementyParteru;
            }
        }
    };

    Garaz[] mojeGaraze = {
        new Garaz() {
            {
                szerokoscBramy = 3;
                wysokoscBramy = 2;
                szerokosc = 5;
                dlugosc = 5;
            }
        }
    };

    Budynek mojBudynek = new Budynek() {
        {
            powierzchniaDzialki = 20;
            pietra = mojePietra;
            garaze = mojeGaraze;
        }
    };

    System.out.println(mojBudynek.PoliczPole());
}
```