

Podstawy programowania

Ćwiczenie 1

Języki programowania obiektowego

Język programowania to formalny system notacji oraz zasad, które są wykorzystywane do wytwarzania programów komputerowych. Służą do przekazywania instrukcji komputerowi w celu wykonania określonych zadań. W przypadku tzw. obiektowych języków programowania, programy są zorganizowane wokół koncepcji obiektów. Podstawowe cechy języków programowania (zwłaszcza obiektowych) obejmują:

- składnię,
- semantykę,
- zmienne, stałe oraz typy danych,
- kontrolę przepływu,
- funkcje oraz obiekty,
- dziedziczenie,
- polimorfizm.

Przykład obiektowych języków programowania, obejmują Java, Python, C++, C# i wiele innych. Języki te, pozwalają programistom tworzyć obiekty, które są instancjami klas, co ułatwia organizację i strukturę kodu w bardziej zrozumiały oraz przyjazny sposób. Programowanie obiektowe promuje koncepty tj. hermetyzacja, abstrakcja, dziedziczenie oraz polimorfizm, które wspierają zasady projektowania oprogramowania. Na potrzeby zajęć, wybrany został język **Java**.

Składnia

Języki programowania mają określoną składnię, która definiuje, jak poprawnie napisać instrukcje w danym języku. Obejmuje ona strukturę kodu, zasady deklaracji zmiennych, stałych, pętli, instrukcji warunkowych oraz wiele innych elementów.

Semantyka

Odnosi się do znaczenia instrukcji w języku programowania. Określa, jakie operacje wykonują różne konstrukcje językowe oraz jakie efekty operacje te, mają na dane.

Zmienne, stałe oraz typy danych

W językach programowania, programiści mogą deklarować zmienne oraz stałe, czyli symboliczne nazwy, które przechowują dane w pamięci komputera. Typy danych definiują rodzaj danych, jakie mogą być przechowywane w zmiennych oraz stałych, tj. liczby całkowite, zmiennoprzecinkowe czy też łańcuchy znaków.

Kontrola przepływu

W kodzie programów, istnieje możliwość kontrolowania przepływu wykonywania kodu w zależności od określonych warunków za pomocą tzw. instrukcji warunkowej (np. if ... else) oraz pętli (np. for lub while).

Funkcje oraz obiekty

Języki programowania (zwłaszcza obiektowe), kod jest zorganizowany w funkcje (zwane też metodami), które grupują pewne operacje. Obiekty natomiast są instancjami klas, które zawierają dane oraz metody związane z tymi danymi.

Dziedziczenie

W obiektowych językach programowania umożliwia jednej klasie (obiektowi) dziedziczenie cech (metod i pól) innej klasy. Pozwala to na ponowne użycie kodu oraz organizację hierarchii klas.

Polimorfizm

Pozwala na stosowanie tych samych interfejsów np. metod, do różnych typów obiektów. Zwiększa to elastyczność oraz modularność kodu.

Java

Java jest obiektowym językiem programowania, stworzonym przez Jamesa Goslinga, Mike'a Sheridana oraz Patricka Naughtona w firmie Sun Microsystems, a obecnie rozwijany jest przez firmę Oracle. Została zaprezentowana w 1995 roku i szybko zdobyła popularność ze względu na swoją przenośność, niezawodność i możliwość używania w różnych obszarach programowania. Kluczowe cechy tego języka to:

- przenośność,
- obiektowość,
- bezpieczeństwo,
- wielowątkowość,
- zarządzanie pamięcią,
- rozległa biblioteka standardowa.

Java jest szeroko stosowana w różnych obszarach, tj. rozwijanie aplikacji mobilnych, tworzenie oprogramowania serwerowego, projektowanie aplikacji internetowych, a także w dziedzinie systemów wbudowanych.

Przenośność

Oznacza, że programy napisane w Javie mogą być uruchamiane na różnych platformach bez konieczności modyfikowania kodu źródłowego. Osiągane jest to dzięki użyciu maszyny wirtualnej Java (JVM), która tłumaczy kod Java na kod zrozumiały dla konkretnego systemu operacyjnego.

Obiektowość

Oznacza, że wszystko w Javie jest reprezentacją obiektu. Programy Java składają się z klas oraz obiektów, co ułatwia organizację i zarządzanie kodem.

Bezpieczeństwo

Java wprowadza szereg funkcji bezpieczeństwa, które pomagają ochronie przed różnymi zagrożeniami, tj. błędy bufora czy ataku typu SQL injection.

Wielowątkowość

Java wspiera programowanie wielowątkowe, co umożliwia równoczesne wykonywanie różnych fragmentów kodu. Jest to szczególnie przydatne w rozwijaniu efektywnych oraz w pełni responsywnych aplikacji.

Zarządzanie pamięcią

Java obsługuje automatyczne zarządzanie pamięcią, co oznacza, że programista nie musi ręcznie alokować oraz zwalniać pamięć komputera. Ułatwia to unikanie wielu błędów związanych z zarządzaniem pamięcią.

Rozległa biblioteka standardowa

Java dostarcza bogatą bibliotekę standardową, która zawiera gotowe rozwiązania dla wielu problemów, co przyspiesza proces programowania.

Integrated Development Environment (IDE) oraz edytor kodu

IDE (Integrated Development Environment) to zintegrowane środowisko programistyczne, które zapewnia programistom narzędzia do tworzenia, testowania, debugowania oraz wdrażania oprogramowania w jednym miejscu. Łączy w sobie różne funkcje tj. edytor kodu, narzędzia do zarządzania projektami, kompilatory, debuggery i inne, aby ułatwić i usprawnić proces tworzenia oprogramowania. IDE dostarcza kompleksowe rozwiązanie dla programistów, umożliwiając im efektywną pracę nad projektem.

Edytory kodu to narzędzia skoncentrowane głównie na edycji kodu źródłowego. Oferuje funkcje tj. podświetlanie składni, numerowanie linii, a także udostępnianie pewnych funkcji, ułatwiających pracę z kodem. Elementy te, nie czynią z nich narzędzi kompleksowych tj. IDE. Są zwykle bardziej minimalistyczne i elastyczne, co pozwala programistom dostosować je do swoich preferencji i potrzeb. Wiele edytorów kodu wymaga od programistów integrowanie różnych narzędzi tj. kompilatory czy też debuggery. Na potrzeby zajęć, przedstawione zostanie narzędzie **Visual Studio Code**, będące edytorem kodu.

Visual Studio Code

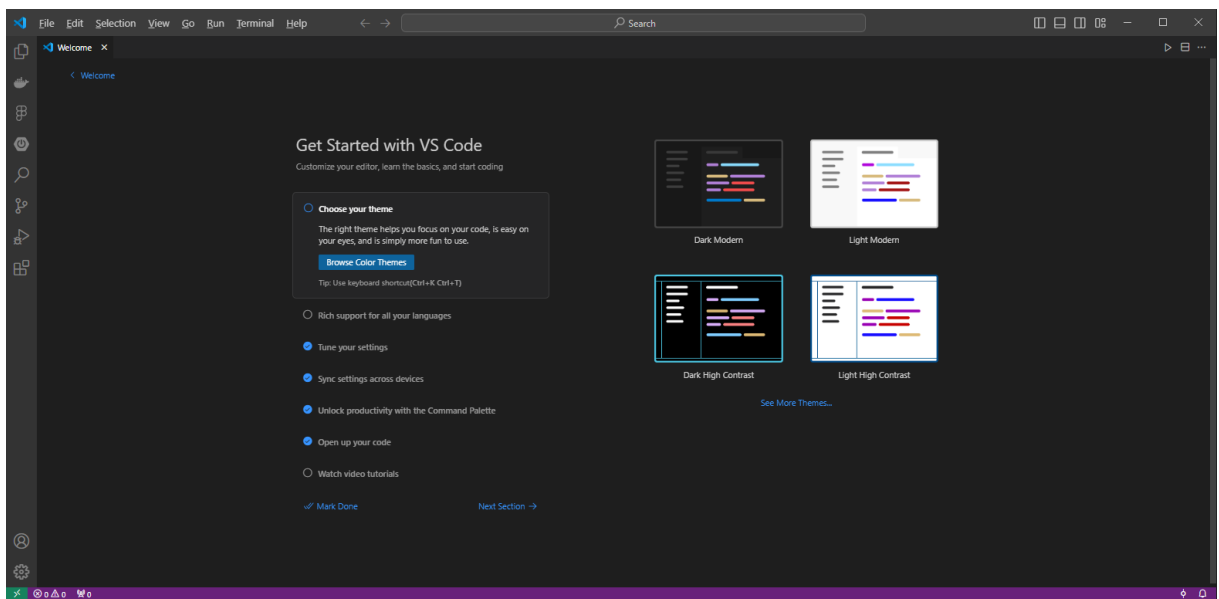
Visual Studio Code (VSC) to bezpłatny, lekki, zaawansowany i elastyczny edytor kodu opracowany przez firmę Microsoft. Narzędzie to, jest popularne wśród programistów do tworzenia różnego rodzaju aplikacji, w tym aplikacji webowych, mobilnych, desktopowych i wiele innych. Zdobył popularność ze względu na swoją prostotę, elastyczność oraz bogatą funkcjonalność. Jest używany przez programistów na różnych platformach (Windows, macOS, Linux). Do korzyści wynikających z jego używania możemy zaliczyć:

- darmowy,
- open source,
- wsparcia dla wielu języków programowania,
- lekki i szybki,
- integracja z systemem kontroli wersji,
- rozszerzenia,
- inteligentne funkcje edytora,
- debugowanie,
- wbudowany terminal,
- współpraca z chmurą.

Instalacja

Aby zainstalować Visual Studio Code (na systemie operacyjnym Windows), należy wykonać następujące kroki:

- wejść na oficjalną stronę VSC (<https://code.visualstudio.com/>),
- na stronie głównej należy kliknąć przycisk do pobrania dla systemu Windows,
- po zakończeniu pobierania, należy uruchomić pobrany plik instalatora,
- po uruchomieniu instalatora, należy postępować zgodnie z instrukcjami na ekranie. Należy zaakceptować domyślne ustawienia, ale można również dostosować instalację według własnych preferencji,
- po zakończeniu instalacji, możemy uruchomić Visual Studio Code z menu Start lub korzystając z utworzonego na pulpicie skrótu.



System kontroli wersji

System kontroli wersji to narzędzie, które umożliwia śledzenie zmian w kodzie źródłowym i zarządzanie historią projektu. Głównym celem systemu kontroli wersji jest umożliwienie programistom efektywnego współpracowania, śledzenia ewolucji projektu oraz przywracania poprzednich wersji kodu w przypadku błędów lub innych problemów. Istnieją dwa główne typy VCS (z ang. Version Control System):

- rozproszony system kontroli wersji (DVCS),
- centralizowany system kontroli wersji (CVCS).

Do podstawowych funkcji systemów kontroli wersji należą:

- zapis historii zmian,
- śledzenie zmian,
- rozgałęzianie i scalanie (tzw. branching oraz merging),
- współpraca zespołowa,

- śledzenie autorstwa.

Do popularnych systemów kontroli wersji należą:

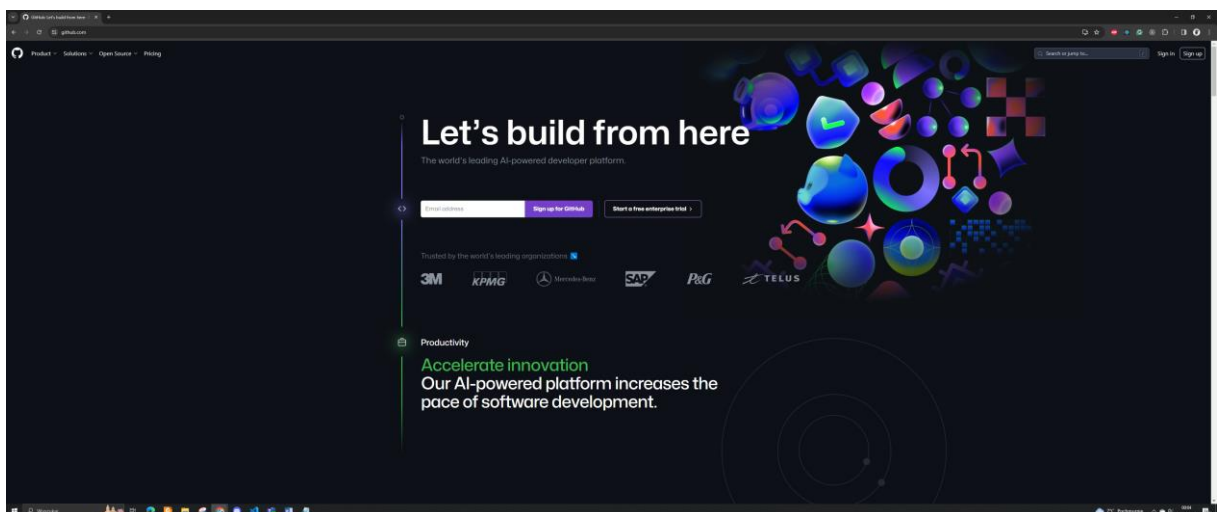
- Git,
- Subversion (SVN),
- Mercurial.

Systemy kontroli wersji to kluczowe narzędzia dla każdego projektu programistycznego, a ich stosowanie przynosi wiele korzyści, tj. lepsza współpraca zespołowa, śledzenie błędów, łatwiejsze utrzymanie i zarządzanie projektem. Na potrzeby zajęć, opisany oraz wykorzystywany będzie system kontroli wersji **Git**.

Github

To platforma internetowa umożliwiająca zarządzanie projektem, śledzenie zmian w kodzie źródłowym oraz współpracę programistyczną. Narzędzie to, oparte jest na systemie kontroli wersji Git, które umożliwia programistom wspólną pracę nad projektem, śledzenie historii zmian, rozwiązywanie konfliktów i wiele innych.

Jest szeroko używany w społeczności programistycznej jako platforma do hostowania otwartoźródłowych projektów (open source), współpracy zespołowej i zarządzania kodem źródłowym. Wprowadza wiele narzędzi ułatwiających pracę programistyczną i wspiera otwarty charakter rozwoju wielu projektów. W celu korzystania z Github należy założyć na nim najpierw konto.

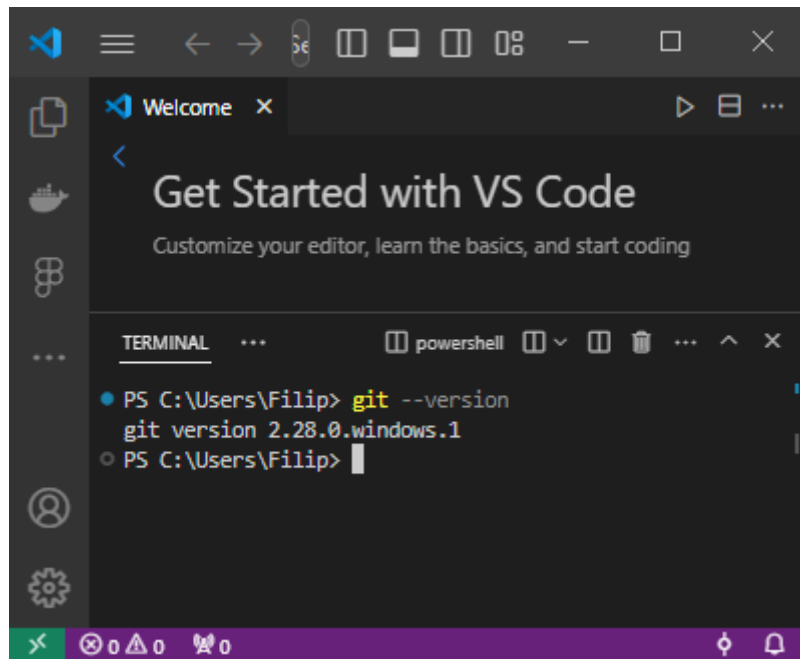


Instalacja

W pierwszej kolejności, należy przejść na oficjalną stronę Git (<https://git-scm.com/>) i pobrać go, za pomocą odpowiedniego przycisku, który pobierze wersję instalatora dla systemu Windows. Po pobraniu, należy uruchomić instalator. Zaproponuje on podczas procesu instalacji domyślne ustawienia, ale można dostosować je według własnych preferencji. Podczas instalacji, istnieje możliwość wyboru edytora tekstu. Dodatkowo, w trakcie instalacji można zaznaczyć opcję – *Git from the command line and also from 3rd-party software*. W celu dodania Git do zmiennej środowiskowej PATH, krok ten jest zalecany. Ułatwi to w przyszłości korzystanie z Git z poziomu wiersza poleceń. Po zakończeniu instalacji, należy w terminalu użyć polecenia:

git --version

Zależnie od otrzymanej reakcji na polecenie, uzyskamy informację czy Git został zainstalowany i jest wykrywany w systemie.



Konfiguracja

Git wymaga prawidłowego skonfigurowania, by połączyć nasze konto Git z urządzeniem na którym wytwarzamy oprogramowanie. W pierwszym kroku, należy ustawić nazwę użytkownika oraz adres e-mail. Możemy tego dokonać za pomocą terminala systemowego (tzw. cmd) lub terminala dostępnego w VSC a następnie, wpisując w terminalu następujące polecenie i podając swoje dane użytkownika:

git config --global user.name „Twoje Imię Nazwisko”

git config --global user.email „twój@email.com”

Jeżeli, nie wybrałeś edytora podczas instalacji, możesz to zrobić później za pomocą polecenia:

git config --global core.editor „code --wait”

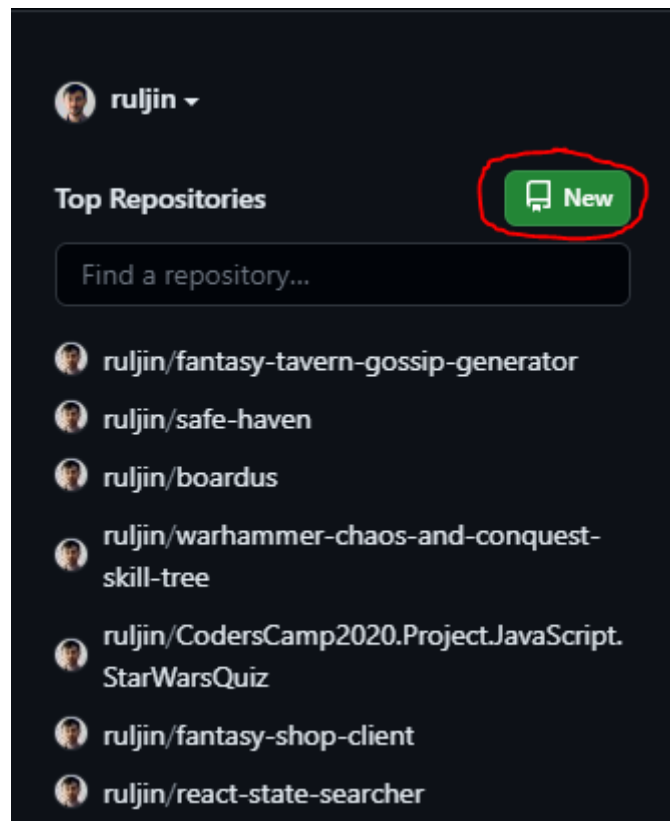
Polecenie to, ustawia VSC jako domyślny edytor.

Integracja z Visual Studio Code

W celu skonfigurowania VSC z Gitem, należy wykonać kilka kroków. Pierwszym, jest uruchomienia rozszerzeń w VSC (zakładka: Extensions) i wyszukania w nich hasła „Git”. Po odnalezieniu tego rozszerzenia, należy je zainstalować. Następnie, gdy otworzysz projekt w VSC i projekt jest w repozytorium Git, VSC automatycznie rozpozna go i zacznie śledzić zmiany. Po tych krokach, Git powinien być zarówno skonfigurowany w systemie Windows jak i zintegrowany z VSC, umożliwiając korzystanie z poleceń Git zarówno z poziomu terminalu, jak i interfejsu graficznego dostępnego w VSC.

Tworzenie nowego repozytorium

Istnieje kilka sposobów tworzenia nowego repozytorium. Jedną z możliwości polega na zalogowaniu się na swoje konto na Github. W następnym kroku, należy kliknąć przycisk new, znajdujący się w lewym, górnym rogu głównej strony po zalogowaniu się.



Następnie, zostaniemy przekierowani na stronę z formularzem, gdzie możemy uzupełnić następujące informacje:

- repository template – czyli szablon repozytorium, jeżeli takowe posiadamy,
- owner – czyli jednostka organizacyjna, która będzie odpowiadać oraz zarządzać repozytorium (ma znaczenie gdy należymy do większej liczby jednostek organizacyjnych, niż nasze własne konto),
- public/private – po wybraniu jednostki organizacyjnej, pojawi się nam lista wyboru mówiąca o dostępności do tworzonego repozytorium,
- repository name – nazwa repozytorium,
- description – opcjonalne pole na krótki opis repozytorium,
- initialize this repository with – możliwość dodania do nowo tworzonego repozytorium, podstawowego pliku README.md, szablonu pliku .gitignore oraz licencji na podstawie, której regulującej co można z treścią tego repozytorium zrobić.

Następnie, po uzupełnieniu formularza, należy kliknąć create repository.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *


Repository name *

Choose an owner ▾

/

Great repository names are short and memorable. Need inspiration? How about [miniature-waffle](#) ?

Description (optional)

 Please choose an owner to see the available visibility options.

Initialize this repository with:

☐

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Create repository

I gotowe. W ten oto sposób, utworzone zostało nowe, puste repozytorium. Na koniec, przeniesieni zostaniemy na ekran, gdzie znajduje się krótki opis w jaki sposób, możemy połączyć nasz lokalny katalog do tego repozytorium z poziomu terminala oraz wypchnąć na niego zmiany, połączyć istniejące repozytorium i wypchnąć zmiany oraz zaimportować kod z innego repozytorium.

The screenshot shows the GitHub interface for a new repository named 'lorem-ipsum'. At the top, there are buttons for 'Pin', 'Unwatch' (1), 'Fork', and 'Star' (0). Below this, there are two main sections: 'Set up GitHub Copilot' and 'Add collaborators to this repository'. The 'Quick setup' section offers options to 'Set up in Desktop', 'HTTPS', or 'SSH' with the URL 'https://github.com/ruljin/lorem-ipsum.git'. It also provides instructions on how to get started by creating a new file or uploading an existing file. The '...or create a new repository on the command line' section shows a series of git commands to initialize a repository, add a README, commit, and push. The '...or push an existing repository from the command line' section shows commands to add a remote, create a branch, and push. The '...or import code from another repository' section mentions that code from Subversion, Mercurial, or TFS can be imported.

lorem-ipsum Public

Pin Unwatch 1 Fork Star 0

Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot

Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH <https://github.com/ruljin/lorem-ipsum.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# lorem-ipsum" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ruljin/lorem-ipsum.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/ruljin/lorem-ipsum.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Odtwarzanie repozytorium

W celu odtworzenia już istniejącego repozytorium, do którego mamy dostęp należy po wejściu na nie, kliknąć code.

The screenshot shows the GitHub interface for a repository named 'character-creator-client'. At the top, there are buttons for 'Pin' and 'Unwatch' (2). Below this, there are buttons for 'main', '1 Branch', and '0 Tags'. A search bar 'Go to file' and an 'Add file' button are also visible. The 'Code' button is highlighted with a red circle. A dropdown menu is open, showing options for 'Local' and 'Codespaces'. Under 'Local', there are options for 'Clone' (with sub-options for 'HTTPS', 'SSH', and 'GitHub CLI'), 'Open with GitHub Desktop', and 'Download ZIP'. The 'Clone' option is selected, and the URL 'https://github.com/ruljin/character-creator-cl...' is shown. The repository content is listed below, including files like 'public', 'src', '.env.example', '.gitignore', 'LICENSE', 'README.md', 'package.json', and 'yarn.lock'.

character-creator-client Public

Pin Unwatch 2

main 1 Branch 0 Tags

Go to file Add file

Code

Local Codespaces

Clone ?

HTTPS SSH GitHub CLI

<https://github.com/ruljin/character-creator-cl...>

Clone using the web URL

Open with GitHub Desktop

Download ZIP

public Feature/structure (#6)

src fix: change size prop

.env.example Feature/restructurin

.gitignore chore: remove .env

LICENSE Initial commit

README.md Create app with cre

package.json Feature/auth (#6) 3 years ago

yarn.lock Create app with create-react-app 3 years ago

Spowoduje to rozwinięcie się listy, gdzie będziemy mieli dostęp do unikalnego linku, potrzebnego do użycia polecenia `git clone`. Następnie, należy otworzyć swój edytor w miejscu, gdzie chcemy repozytorium zdalne odtworzyć. Ostatni krok, polega na wpisaniu w terminalu VSC polecenia:

`git clone https://github.com/nazwa_uzytkownika/nazwa_repozytorium`

Polecenie to, odtworzy nasze repozytorium zdalne w miejscu, gdzie otwarty jest edytor i/lub gdzie wskazuje aktualnie terminal (jeżeli zmienialiśmy lokalizację za jego pomocą).

Podstawowe polecenia Git

Istnieje wiele poleceń w Git. Niżej, wymienione zostało kilka z nich:

- **`git init`** – inicjalizacja nowego repozytorium,
- **`git status`** – sprawdzenie stanu zmian,
- **`git add nazwa_pliku`** – dodanie plików do obszaru *stage*,
- **`git add *`** - dodanie wszystkich plików do obszaru *stage*,
- **`git commit -m „opis zmian”`** – zapisanie zmian w lokalnym repozytorium,
- **`git branch nazwa_galezi`** – utworzenie nowej gałęzi,
- **`git checkout nazwa_galezi`** – przełączenie się na inną gałąź,
- **`git checkout -b nazwa_galezi`** – utworzenie nowej gałęzi oraz przełączenie się na nią,
- **`git push origin nazwa_galezi`** – wysłanie zmian do repozytorium zdalnego.

Metody uruchamiania aplikacji w języku Java

W Java, aplikacje są uruchamiane poprzez interpreter maszyny wirtualnej Java (Java Virtual Machine, czyli JVM). Aby uruchomić program napisany w Java, należy skompilować kod źródłowy do postaci bajtowej (pliki `.class`), a następnie uruchomić go przy użyciu JVM. Można dokonać tego na kilka sposobów:

- użyć polecenia `java` w terminalu,
- użycie polecenia `javac` oraz `java` w jednym kroku w terminalu,
- użycie IDE,
- użycie `jar`.

Kompilacja i uruchamianie w konsoli

Stwórz na początek plik **example.java** w VSC a następnie, przekopiuj do niego podany niżej fragment kodu:

```
public class example {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

W pierwszym kroku, należy skompilować pliki źródłowe do plików bajtowych za pomocą polecenia:

javac example.java

Gdzie **example** należy w przyszłości zastąpić nazwą interesującego nas pliku. Gdy zrobimy to, gdy użyjemy polecenia:

java example

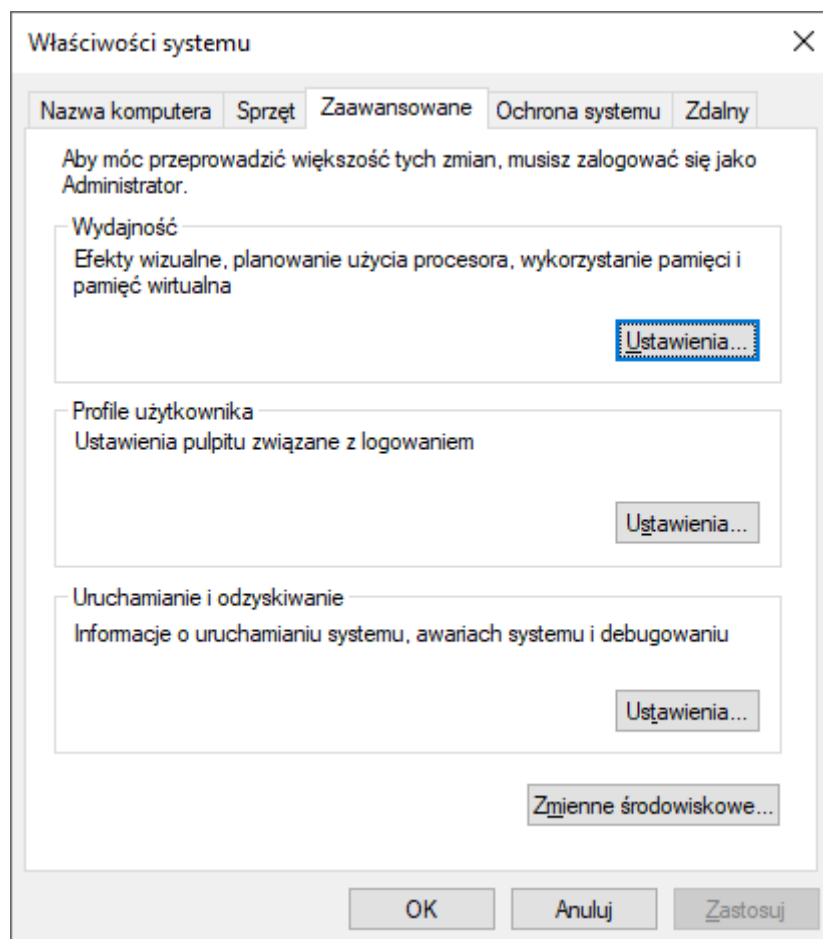
Nasz kod powinien zostać wykonany. Polecenia można oczywiście łączyć, co oznacza, że skrócona wersja tych poleceń, wyglądała by następująco:

javac *.java ; java example

Proces kompilacji zostaje tu skrócony dzięki symbolowi gwiazdki (*) a samo polecenie, zredukowanego do jednego. Ostatnia metoda polega na utworzeniu tzw. pliku jar. W tym celu należy tak jak na początku użyć polecenia **javac** a następnie użyć polecenia:

jar cvfe example.jar example example.class

Jeżeli polecenie wyrzuci błąd, oznacza to, że polecenie nie jest rozpoznawane przez terminal. Najpewniej przyczyną jest brak dostępu w bieżącej lokalizacji lub nie uwzględnienie go w zmiennej środowiskowej PATH. W tym celu należy wyszukać i otworzyć w systemie operacyjnym „Edytuj zmienne środowiskowe systemu” a następnie kliknąć „Zmienne środowiskowe”.



Należy odnaleźć tam w „Zmienne systemowe” – PATH i edytować je. Należy dodać tam ścieżkę, gdzie znajduje się katalog bin naszego zainstalowanego Java JDK, np.:

C:\Program Files\Java\jdk-21\bin

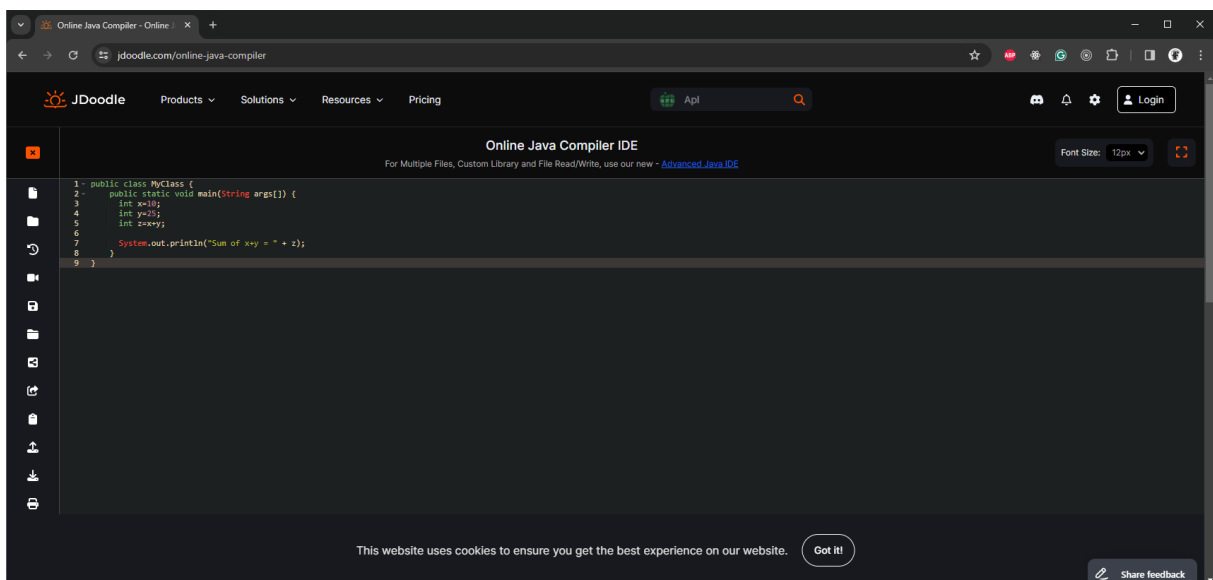
Dodanie nowej wartości do PATH wymaga zrestartowanie wszystkich aktywnych terminali (w tym uruchomienie ponowne VSC). Następnie, polecenie powinno już działać. W celu uruchomienia projektu w postaci jar, należy użyć polecenia:

java -jar example.jar

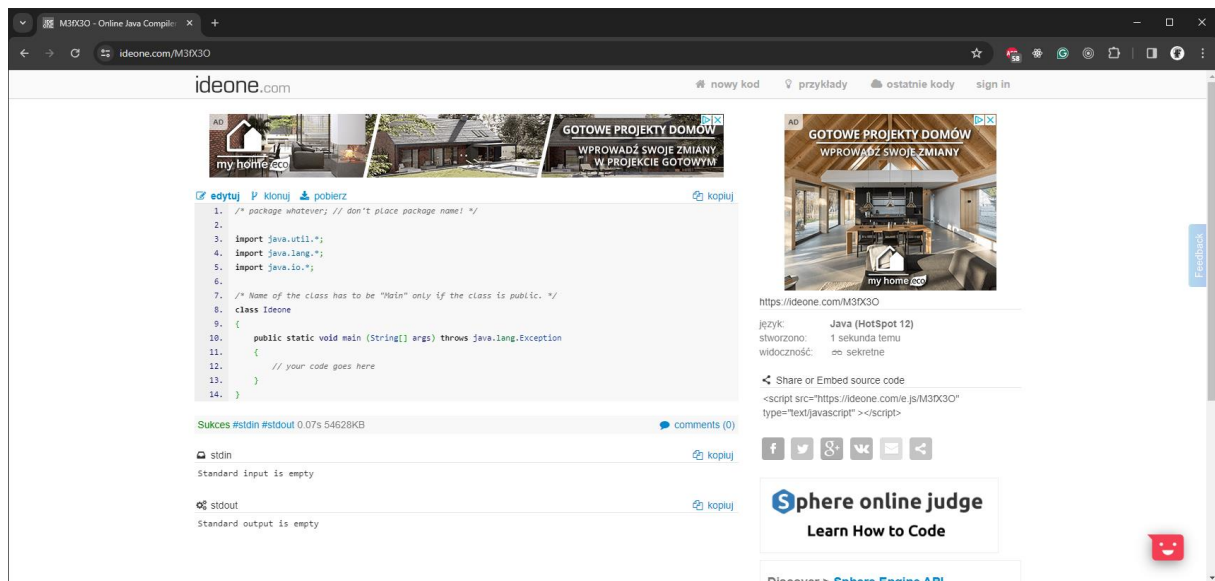
Kompilacja w środowiskach internetowych

Kompilacja kodu źródłowego w środowiskach internetowych może być realizowana na różne sposoby, w zależności od dostępnego narzędzia i środowiska programistycznego. Poniżej, przedstawione zostaną dwie:

- użycie środowisk online (np. JDoodle) – istnieją platformy online, które umożliwiają kompilację kodu źródłowego w wielu językach programowania bez konieczności instalacji narzędzi na lokalnej maszynie. Pozwalają na wprowadzanie kodu, kompilację i uruchamianie programów), bezpośrednio w przeglądarce.



- użycie narzędzi online do kompilacji i wykonania (np. ideone) – niektóre strony internetowe, oferują narzędzia do kompilacji i wykonania kodu w konkretnych językach programowania. Na takich platformach można przekopiować swój kod, wybrać język programowania i uruchomić go online.



Ćwiczenia

1 – Uruchamianie w konsoli

Napisz prosty program w Javie, który wyświetli Twoje imię i nazwisko oraz numer albumu i uruchom go z konsoli.

2 – Testowanie kompilacji

Do napisanego wcześniej programu dodaj kilka klas i przetestuj kompilację.

3 – Uruchamianie programu Java online

Wykorzystaj dowolną platformę internetową do napisania i uruchomienia programu Java online.