# Politechnika Świętokrzyska w Kielcach

**Wydział Elektrotechniki, Automatyki i Informatyki**
**Katedra Informatyki, Elektroniki i Elektrotechniki**

| Kierunek<br>**Informatyka** | Projekt<br>**Programowanie Obiektowe 2 - Java** | |
|---|---|---|
| Grupa dziekańska<br>**2ID12A** | Temat projektu<br>Gra – Tetris | Wykonali:<br>**Zając Kamil**<br>**Głód Jakub** |

# Procentowy wkład pracy włożonej w stworzenie i rozwój projektu poszczególnych członków:

- Głód Jakub (50%)
- Zając Kamil(50%)

# Opis projektu

**Projekt to prosta implementacja gry Tetris w języku Java, wykorzystując bibliotekę graficzną JFrame. W grze, elementy Tetrisa (bloki) spadają z góry ekranu, a gracz ma za zadanie ułożyć je w linie poziome, które znikają, przynosząc punkty. Projekt korzysta z koncepcji okien graficznych JFrame do przedstawienia interfejsu gry.**

**Main:**

## 1) Informacje na temat funkcjonalności projektu:

- Gra Tetris: Zawiera podstawową mechanikę gry Tetris, gdzie bloki spadają z góry ekranu, a gracz może poruszać i obracać je w celu ułożenia w pełne linie.

## 2) Informacje na temat sposobu uruchomienia oraz obsługi projektu:

- Uruchamianie: Projekt jest aplikacją konsolową. Po uruchomieniu, pojawi się okno gry Tetris.

**3) Informacje na temat stworzonych klas, metod, funkcji (bez kodu źródłowego) z 5) opisem ich podstawowej funkcjonalności oraz ich przeznaczeniem:**

- Main Class (Main):
    - main(String[] args): Metoda główna, inicjuje okno JFrame, ustawia parametry gry, tworzy panel gry (GamePanel) i uruchamia grę.

**4) Opis stworzonych klas, metod, funkcji:**

- GamePanel Class (GamePanel):
    - launchGame(): Rozpoczyna główną pętlę gry, obsługując logikę gry, ruchy bloków, kolizje itp.
- JFrame Class:
    - setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE): Ustawia domyślne zachowanie dla zamknięcia okna.
    - setResizable(false): Uniemożliwia zmianę rozmiaru okna.
    - add(Component comp): Dodaje komponent do kontenera.
    - pack(): Dostosowuje rozmiar okna do preferowanego rozmiaru jego podkomponentów.
    - setLocationRelativeTo(null): Ustawia położenie okna na środku ekranu.
    - setVisible(true): Ustawia widoczność okna.

# Game Panel:

## 1) Ogólny opis klasy GamePanel:

Klasa GamePanel pełni rolę panelu gry w aplikacji Tetris. Jest rozszerzeniem klasy JPanel z biblioteki graficznej Swing. Odpowiada za obsługę rysowania elementów gry, zarządzanie logiką gry i interakcją z użytkownikiem.

## 2) Informacje na temat funkcjonalności klasy GamePanel:

- Rysowanie: Zarządza rysowaniem elementów gry na ekranie.
- Logika gry: Obsługuje logikę gry, taką jak ruchy bloków, kolizje, punktację itp.
- Interakcja z klawiaturą: Reaguje na interakcje klawiatury, takie jak poruszanie blokami, pauzowanie gry.
- Obsługa wątku: Uruchamia grę jako osobny wątek (gameThread), co umożliwia odświeżanie i aktualizację gry z określoną częstotliwością.

## 3) Informacje na temat sposobu uruchomienia oraz obsługi klasy GamePanel:

- Konstruktor: Inicjalizuje ustawienia panelu, takie jak rozmiar, kolor tła, obsługa klawiatury. Tworzy obiekt klasy PlayManager (pm), który zarządza logiką gry.

- launchGame(): Rozpoczyna główny wątek gry (gameThread), co umożliwia płynne odświeżanie i aktualizację gry.

**4) Informacje na temat stworzonych metod klasy GamePanel:**

- run(): Metoda z interfejsu Runnable, implementuje główną pętlę gry. Monitoruje czas, aby utrzymać stałą ilość klatek na sekundę (FPS).
- update(): Aktualizuje logikę gry, jeżeli gra nie jest wstrzymana (nie jest włączony tryb pauzy i nie wystąpił koniec gry).
- paintComponent(Graphics g): Przesłonięta metoda z klasy JPanel, rysuje elementy gry na panelu.

**5) Opis stworzonych klas, metod, funkcji:**

- GamePanel korzysta z klasy PlayManager, która prawdopodobnie zarządza logiką gry Tetris, taką jak obsługa kształtów bloków, kolizje itp.
- run(): Główna pętla gry, która monitoruje czas, aktualizuje logikę gry, i rysuje elementy gry.
- update(): Metoda, która wywołuje aktualizację logiki gry, jeżeli warunki umożliwiają aktualizację (brak pauzy i brak zakończenia gry).
- paintComponent(Graphics g): Metoda rysująca elementy gry na panelu.

## Klawiatura:

### 1) Ogólny opis klasy Klawiatura:

Klasa Klawiatura obsługuje zdarzenia związane z klawiaturą w grze Tetris. Implementuje interfejs KeyListener w celu przechwytywania zdarzeń klawiatury.

### 2) Informacje na temat funkcjonalności klasy Klawiatura:

- Przechwytywanie klawiszy: Odpowiada za przechwytywanie zdarzeń naciśnięcia klawiszy strzałek (góra, dół, lewo, prawo) oraz klawisza Esc.
- Ustawianie flag: Ustawia odpowiednie flagi (upPressed, downPressed, leftPressed, rightPressed, pausePressed) w zależności od klawiszy naciśniętych przez użytkownika.

### 3) Informacje na temat sposobu użycia i obsługi klasy Klawiatura:

- Implementacja interfejsu: Klasa implementuje interfejs KeyListener, co umożliwia reakcję na zdarzenia związane z klawiaturą.
- Metoda keyPressed(KeyEvent e): Reaguje na zdarzenia naciśnięcia klawiszy, ustawiając odpowiednie flagi w zależności od naciśniętego klawisza.

- Metoda keyReleased(KeyEvent e): Pusta implementacja, nie jest używana w tym kontekście.
- Metoda keyTyped(KeyEvent e): Pusta implementacja, nie jest używana w tym kontekście.

**4) Informacje na temat stworzonych zmiennych w klasie Klawiatura:**

- upPressed, downPressed, leftPressed, rightPressed, pausePressed: Zmienne flagowe określające, czy odpowiednie klawisze są naciśnięte. Są one ustawiane na true lub false w zależności od akcji użytkownika.

**5) Opis stworzonych klas, metod, funkcji:**

- Klasa Klawiatura nie korzysta z innych klas ani metod, ale dostarcza mechanizm przechwytywania i obsługi zdarzeń klawiatury w kontekście gry Tetris.
- Zmienne flagowe upPressed, downPressed, leftPressed, rightPressed, pausePressed są używane do informowania innych części programu o tym, które klawisze są aktualnie naciśnięte.

**PlayManager:**

## Ogólny opis klasy PlayManager:

Klasa PlayManager odpowiada za zarządzanie rozgrywką w grze Tetris. Kontroluje aktualny stan gry, poziom, ilość linii, wynik oraz obsługuje mechanikę dodawania nowych bloków Tetris i sprawdzania warunków zakończenia gry.

## 1) Informacje na temat funkcjonalności klasy PlayManager:

- **Zarządzanie planszą gry:**
  - Ustala granice planszy na podstawie szerokości i wysokości.
  - Określa obszar, na którym poruszają się bloki Tetris.
- **Inicjalizacja Tetris Blocks:**
  - Tworzy i ustawia pierwszy blok Tetris na planszy startowej.
  - Przechowuje informacje o następnym bloku Tetris do pojawienia się.
- **Aktualizacja gry:**
  - Sprawdza, czy aktualny blok Tetris jest aktywny, jeśli nie, przenosi go do listy statycznych bloków.
  - Ustawia nowy blok Tetris jako aktualny.
  - Sprawdza warunki zakończenia gry.
  - Aktualizuje listę statycznych bloków oraz sprawdza i usuwa pełne linie.

- **Wybór bloku Tetris:**
  - Losowo wybiera jeden z siedmiu rodzajów bloków Tetris.
- **Poziomy, punkty i tempo gry:**
  - Śledzi aktualny poziom, liczbę zdobytych linii i łączny wynik.
  - Zwiększa poziom i przyspiesza tempo gry co 10 zdobytych liniach.

## 2) Informacje na temat sposobu użycia i obsługi klasy PlayManager:

- **Konstruktor:**
  - Ustala początkowe wartości granic planszy, pozycji startowej Tetris Blocks oraz obszaru dla następnego bloku Tetris.
  - Inicjalizuje poziom, liczbę linii, wynik i interwał spadania bloków.
- **Metoda update():**
  - Aktualizuje stan gry, sprawdza zakończenie rundy, przenosi bloki do listy statycznych bloków oraz przygotowuje nowy blok Tetris.
- **Metoda checkDelete():**
  - Sprawdza, czy pełne linie są dostępne do usunięcia, a następnie usuwa i aktualizuje planszę.
- **Metoda pickTetrisBlock():**
  - Losowo wybiera blok Tetris spośród siedmiu rodzajów.

- **Metoda draw(Graphics2D g2):**
    - Rysuje elementy gry, takie jak plansza, obszar dla następnego bloku, poziom, ilość linii, wynik, aktualny blok Tetris, następny blok Tetris oraz statyczne bloki.
    - Informuje o zakończeniu gry lub pauzie.

**3) Informacje na temat stworzonych zmiennych w klasie PlayManager:**

- **WIDTH**, **HEIGHT**: Stałe określające szerokość i wysokość planszy.
- **left_x**, **right_x**, **top_y**, **bottom_y**: Granice obszaru dla bloków Tetris oraz planszy.
- **currentTetrisBlock**, **nextTetrisBlock**: Obiekty reprezentujące aktualny i następny blok Tetris.
- **staticBlocks**: Lista bloków Tetris, które stały się statyczne.
- **level**, **lines**, **score**: Zmienne śledzące poziom, liczbę zdobytych linii i wynik.
- **dropInterval**: Interwał spadania bloków.
- **gameOver**: Flaga informująca o zakończeniu gry.

**4) Opis stworzonych klas, metod, funkcji:**

- Klasa korzysta z klas bloków Tetris (**TetrisBlock_L1**, **TetrisBlock_L2**, ..., **TetrisBlock_Z2**) oraz z klas związanych z rysowaniem (**Graphics2D**, **Color**, **Font**, ...).
- Metoda **pickTetrisBlock()** wybiera losowo jeden z siedmiu rodzajów bloków Tetris.

- Metoda **draw(Graphics2D g2)** rysuje wszystkie elementy gry na ekranie.

**TetrisBlock:**

**1) Ogólny opis klasy TetrisBlock:**

Klasa TetrisBlock reprezentuje ogólną strukturę bloku Tetris. Odpowiada za zarządzanie położeniem, kolizjami oraz ruchem bloku na planszy. Jest klasą bazową dla konkretnych typów bloków Tetris.

**2) Informacje na temat funkcjonalności klasy TetrisBlock:**

- Inicjalizacja i tworzenie bloku:
  - Tworzy tablice obiektów Block reprezentujących poszczególne segmenty bloku.
  - Inicjalizuje zmienne takie jak autoDropCounter, direction, active, deactivating, deactivateCounter.
  - Udostępnia metody do ustawiania współrzędnych bloku, sprawdzania kolizji oraz aktualizacji położenia.
- Aktualizacja położenia bloku:
  - Sprawdza kolizje z innymi blokami oraz krawędziami planszy.
  - Odpowiada za ruch bloku w dół, w lewo i w prawo.

    - Automatycznie opada blok w dół z zadanym interwałem.
- Sprawdzanie kolizji:
    - Sprawdza kolizje z innymi blokami (statycznymi) na planszy.
    - Kontroluje kolizje z krawędziami planszy, co uniemożliwia wychodzenie poza granice.
- Zmiana kierunku bloku:
    - Udostępnia metody do zmiany kierunku bloku zgodnie z zasadami gry Tetris.
- Deaktywacja bloku:
    - Oznacza blok jako nieaktywny, gdy zachodzi kolizja na dole planszy.
    - Przechodzi w tryb deaktywacji, który po krótkim czasie sprawdza, czy blok może być zdezaktywowany.
- Rysowanie bloku:
    - Udostępnia metodę do rysowania bloku na ekranie.

**3) Informacje na temat sposobu użycia i obsługi klasy TetrisBlock:**

- Konstruktor:
    - Tworzy obiekty reprezentujące poszczególne segmenty bloku Tetris.
    - Inicjalizuje zmienne kontrolujące ruch, kolizje i aktywność bloku.
- Metoda update():

    - Aktualizuje położenie bloku na podstawie wprowadzonych zmian.
    - Sprawdza kolizje i reaguje na ruch gracza.
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
    - Odpowiadają za zmianę kierunku bloku zgodnie z zasadami gry Tetris.
- Metoda checkMovementCollision(), checkRotationCollision(), checkStaticBlockCollision():
    - Sprawdzają kolizje z krawędziami planszy, statycznymi blokami i innymi segmentami bloku Tetris.
- Metoda deactivating():
    - Przechodzi w tryb deaktywacji bloku i sprawdza, czy może zostać zdezaktywowany.
- Metoda draw(Graphics2D g2):
    - Rysuje poszczególne segmenty bloku na ekranie.

**4) Informacje na temat stworzonych zmiennych w klasie TetrisBlock:**

- b: Tablica przechowująca obiekty Block reprezentujące poszczególne segmenty bloku Tetris.
- tempB: Tablica pomocnicza, używana do sprawdzania kolizji i aktualizacji położenia.
- autoDropCounter: Licznik kontrolujący automatyczny spadek bloku.
- direction: Zmienna przechowująca kierunek bloku.

- leftCollision, rightCollision, bottomCollision: Flagi informujące o kolizjach z krawędziami planszy.
- active: Flaga informująca o aktywności bloku.
- deactivating: Flaga informująca o trybie deaktywacji bloku.
- deactivateCounter: Licznik kontrolujący czas deaktywacji bloku.

**5) Opis stworzonych klas, metod, funkcji:**

- Korzysta z klasy Block reprezentującej pojedynczy segment bloku Tetris.
- Korzysta z klas związanych z grą Tetris, takich jak Klawiatura i PlayManager.
- Udostępnia metody do ustawiania współrzędnych, zmiany kierunku, aktualizacji położenia i rysowania bloku Tetris.

**Block:**

**1) Ogólny opis klasy Block:**

Klasa Block reprezentuje pojedynczy segment bloku w grze Tetris. Jest to prostokąt o stałym rozmiarze, przechowujący informacje o swoich współrzędnych, kolorze i potrafiący narysować się na ekranie.

## 2) Informacje na temat funkcjonalności klasy Block:

- Inicjalizacja bloku:
  - Przechowuje informacje o współrzędnych x i y, kolorze c oraz stałym rozmiarze bloku (SIZE).
- Rysowanie bloku:
  - Udostępnia metodę draw(Graphics2D g2), która rysuje prostokąt o określonych współrzędnych, kolorze i rozmiarze.

## 3) Informacje na temat sposobu użycia i obsługi klasy Block:

- Konstruktor:
  - Przyjmuje jako argument kolor bloku.
  - Inicjalizuje pole c (kolor) oraz stały rozmiar bloku (SIZE).
- Metoda draw(Graphics2D g2):
  - Rysuje prostokąt o określonych współrzędnych x i y, kolorze c oraz rozmiarze SIZE.
  - Używa marginesu, aby zostawić miejsce na obramowanie bloku.

**4) Informacje na temat stworzonych zmiennych w klasie Block:**

- x, y: Współrzędne bloku, określające pozycję na planszy.
- SIZE: Stały rozmiar bloku.
- c: Kolor bloku.

**5) Opis stworzonych klas, metod, funkcji:**

- Klasa zawiera podstawowe informacje i funkcje potrzebne do reprezentacji pojedynczego segmentu bloku w grze Tetris.
- Udostępnia metodę draw, która pozwala na narysowanie bloku na ekranie.

### 1) Klasa TetrisBlock_Bar:

- Opis ogólny:
  - Reprezentuje blok w kształcie "paska" w grze Tetris, składający się z czterech segmentów ułożonych poziomo.
- Konstruktor:
  - Inicjalizuje obiekt klasy, ustawiając kolor bloku na cyan przy użyciu create(Color.cyan).
- Metoda setXY(int x, int y):
  - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie "paska".
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
  - Określają nowe współrzędne bloków w zależności od kierunku ruchu, umożliwiając obroty bloku.

### 2) Klasa TetrisBlock_L1:

- Opis ogólny:
  - Reprezentuje blok w kształcie litery "L" w grze Tetris, składający się z czterech segmentów ułożonych w kształt litery L.
- Konstruktor:
  - Inicjalizuje obiekt klasy, ustawiając kolor bloku na orange przy użyciu create(Color.orange).

- Metoda setXY(int x, int y):
    - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie litery "L".
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
    - Określają nowe współrzędne bloków w zależności od kierunku ruchu, umożliwiając obroty bloku.

**3) Klasa TetrisBlock_L2:**

- Opis ogólny:
    - Reprezentuje blok w kształcie litery "L" w grze Tetris, składający się z czterech segmentów ułożonych w odwróconą literę L.
- Konstruktor:
    - Inicjalizuje obiekt klasy, ustawiając kolor bloku na blue przy użyciu create(Color.blue).
- Metoda setXY(int x, int y):
    - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie odwróconej litery "L".
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
    - Określają nowe współrzędne bloków w zależności od kierunku ruchu, umożliwiając obroty bloku.

### 4) Klasa TetrisBlock_Square:

- Opis ogólny:
  - Reprezentuje blok w kształcie kwadratu w grze Tetris, składający się z czterech segmentów ułożonych w kwadrat.
- Konstruktor:
  - Inicjalizuje obiekt klasy, ustawiając kolor bloku na yellow przy użyciu create(Color.yellow).
- Metoda setXY(int x, int y):
  - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie kwadratu.
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
  - Brak implementacji, ponieważ kwadrat nie ulega zmianom podczas obrotu.

### 5) Klasa TetrisBlock_T:

- Opis ogólny:
  - Reprezentuje blok w kształcie litery "T" w grze Tetris, składający się z czterech segmentów ułożonych w kształt litery T.
- Konstruktor:
  - Inicjalizuje obiekt klasy, ustawiając kolor bloku na magenta przy użyciu create(Color.magenta).

- Metoda setXY(int x, int y):
    - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie litery "T".
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
    - Określają nowe współrzędne bloków w zależności od kierunku ruchu, umożliwiając obroty bloku.

**6) Klasa TetrisBlock_Z1:**

- Opis ogólny:
    - Reprezentuje blok w kształcie litery "Z" w grze Tetris, składający się z czterech segmentów ułożonych w jedną odmianę litery Z.
- Konstruktor:
    - Inicjalizuje obiekt klasy, ustawiając kolor bloku na red przy użyciu create(Color.red).
- Metoda setXY(int x, int y):
    - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie litery "Z".
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
    - Określają nowe współrzędne bloków w zależności od kierunku ruchu, umożliwiając obroty bloku.

### 7) Klasa TetrisBlock_Z2:

- Opis ogólny:
  - Reprezentuje blok w kształcie litery "Z" w grze Tetris, składający się z czterech segmentów ułożonych w drugą odmianę litery Z.
- Konstruktor:
  - Inicjalizuje obiekt klasy, ustawiając kolor bloku na green przy użyciu create(Color.green).
- Metoda setXY(int x, int y):
  - Ustawia współrzędne bloków w zależności od współrzędnych przekazanych jako argumenty, tworząc blok w kształcie litery "Z".
- Metody getDirection1(), getDirection2(), getDirection3(), getDirection4():
  - Określają nowe współrzędne bloków w zależności od kierunku ruchu, umożliwiając obroty bloku