Programowanie obiektowe

Język Java – część 2
(przykładowa aplikacja)

Paweł Rogaliński
Instytut Informatyki, Automatyki i Robotyki
Politechniki Wrocławskiej

pawel.rogalinski@pwr.wroc.pl

Zamiast wprowadzenia

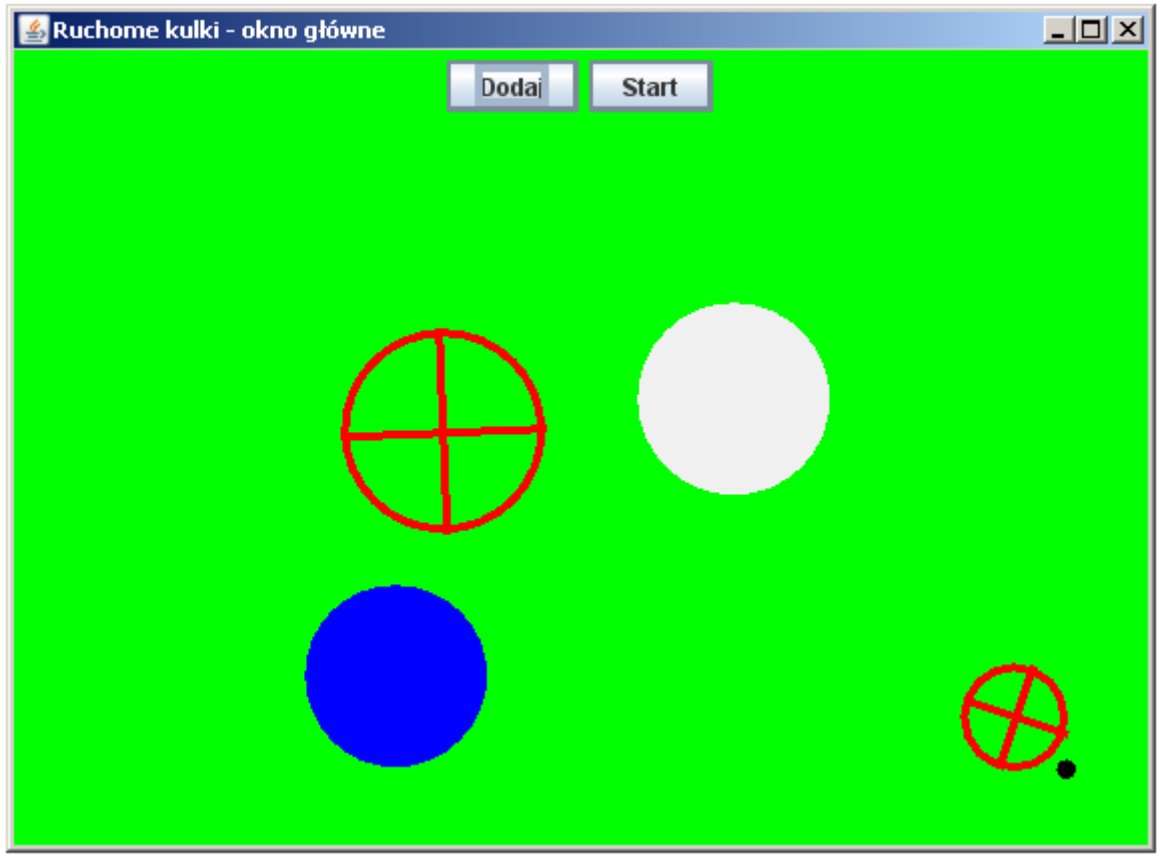
Dla kogo jest dzisiejszy wykład?

- Dla osób, które znają język C/C++ ?
- Dla osób które zaczynają swoją przygodę z językiem Java?
- Dla osób zainteresowanych tworzeniem aplikacji dla systemu Android ?

Czy można **nauczyć** (**nauczyć się**) programowania w języku Java w kilka godzin?

Spróbujmy napisać prostą aplikację graficzną z elementami animacji i interakcji użytkownika!

Zamiast wprowadzenia



Spróbujmy napisać prostą aplikację graficzną z elementami animacji i interakcji użytkownika!

Krok 0: Prosta aplikacja

Kod źródłowy: **RuchomeKulki_00.java**

Cel: Utworzenie aplikacji w języku Java.

- zdefiniowanie klasy ***`RuchomeKulki`***,
- zdefiniowanie statycznej metody ***`main()`***.

Zagadnienia: definicja klasy, metody statyczne, metoda ***`main()`***.

Krok 1: Okno graficzne

Kod źródłowy: **RuchomeKulki_01.java**

Cel: Wyświetlenie okna graficznego.

- importowanie klas z pakietu ***`javax.swing`***,
- dziedziczenie klasy ***`JFrame`***,
- utworzenie konstruktora klasy ***`RuchomeKulki`***,
- utworzenie obiektu reprezentującego okno graficzne.

Zagadnienia: dziedziczenie, konstruktor, wywołanie konstruktora klasy bazowej.

Krok 2: Panel rysunkowy

Kod źródłowy: **RuchomeKulki_02.java**

Cel: Utworzenie panelu reprezentującego rysunek.

- importowanie klas z pakietu ***`java.awt`***,
- utworzenie klasy ***`Rysunek`*** dziedziczącej po klasie ***`JPanel`***,
- utworzenie konstruktora,
- przedefiniowane metody ***`paintComponent`***,
- utworzenie obiektu reprezentującego rysunek w oknie graficznym,
- umieszczenie rysunku w oknie graficznym.

Zagadnienia: przedefiniowanie metody, wywołanie metody z klasy bazowej

Krok 3: Figura

Kod źródłowy: **RuchomeKulki_03.java**

Cel: Definicja klasy reprezentującej figury na rysunku.

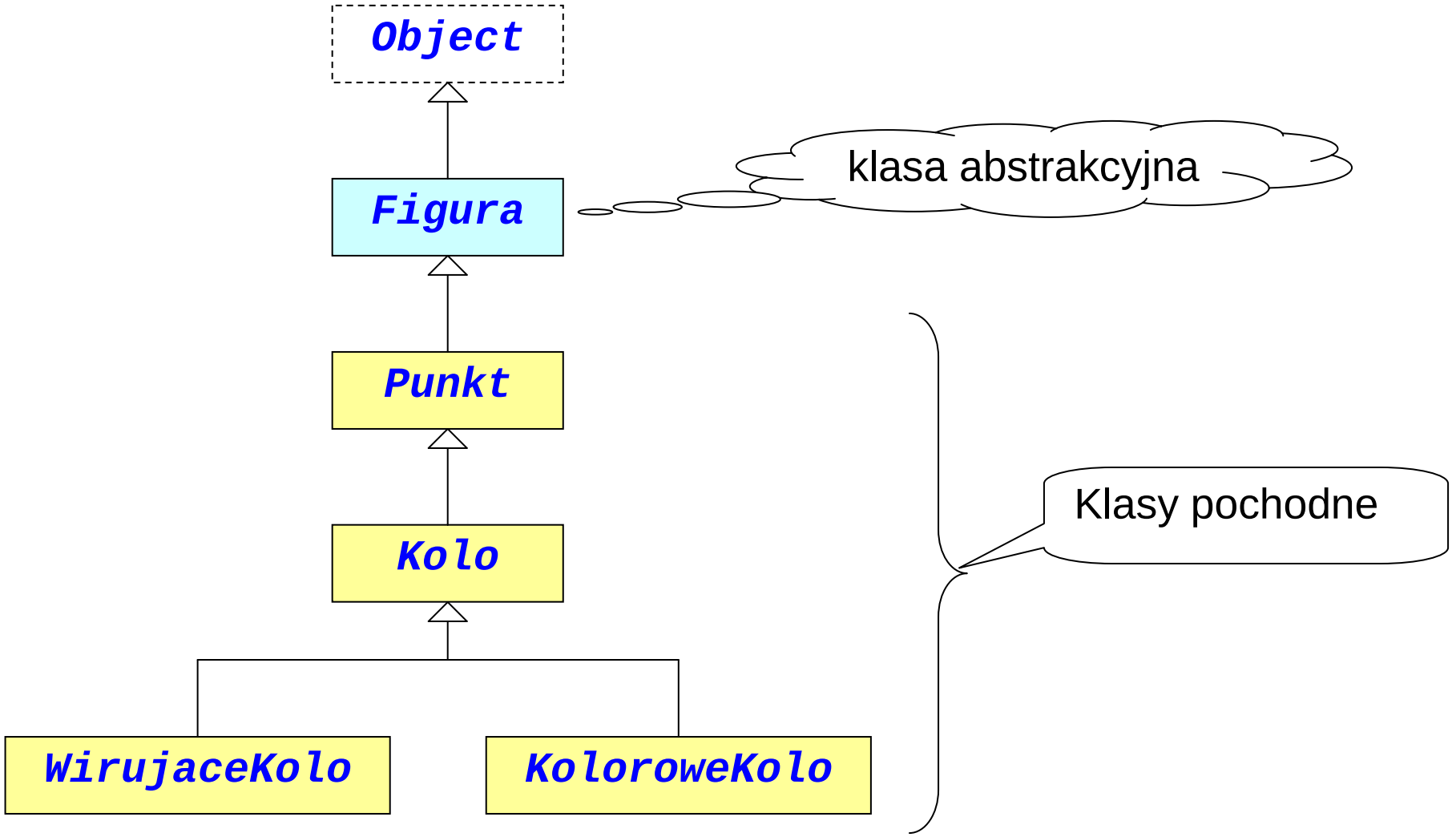
- definicja klasy ***`Figura`***,
- utworzenie metody ***`rysuj`***,
- utworzenie obiektu reprezentującego figurę geometryczną na rysunku,
- wyświetlenie figury w metodzie ***`paintComponent`***,

Zagadnienia: wyświetlanie obiektów w panelu rysunku.

Krok 4: Klasy reprezentujące różne figury

Kod źródłowy: **RuchomeKulki_04.java**

Cel: Utworzenie hierarchii klas reprezentujących różne rodzaje figur.



Krok 4: Klasy reprezentujące różne figury

Kod źródłowy: **RuchomeKulki_04.java**

Cel: Utworzenie hierarchii klas reprezentujących różne rodzaje figur.

- definicja abstrakcyjnej klasy ***`Figura`***,
- definicja klas pochodnych i implementacja metody abstrakcyjnej ***`rysuj`***,
- utworzenie obiektów reprezentujących przykłady różnych figur geometrycznych,
- wyświetlenie wszystkich figur w metodzie ***`paintComponent`***,

Zagadnienia: klasa abstrakcyjna, metoda abstrakcyjna, dziedziczenie pól i metod, implementacja metod abstrakcyjnych, tworzenie obiektów.

Krok 5: Kolekcja figur

Kod źródłowy: **RuchomeKulki_05.java**

Cel: Utworzenie listy do przechowywania wszystkich figur na rysunku.

- importowanie klas z pakietu ***`java.util`***,
- utworzenie kolekcji ***`listaFigur`*** z klasy ***`ArrayList`*** do przechowywania obiektów reprezentujących figury na rysunku,
- próbne utworzenie i dodanie do kolekcji ***`listaFigur`*** obiektów reprezentujących przykłady różnych figur geometrycznych,
- modyfikacja wyświetlenia wszystkich figur w metodzie ***`paintComponent`***.

Zagadnienia: klasy sparametryzowane, pakiet **Java Collections Framework**,
rodzaje kolekcji, dodawanie obiektów do kolekcji,
przeglądanie obiektów z kolekcji.

Krok 6: Animacja ruchu figur

Kod źródłowy: **RuchomeKulki_06.java**

Cel: Animacja ruchu wszystkich figur w obrębie rysunku.

- uzupełnienie metod abstrakcyjnych w klasie ***Figura*** o dodatkowe metody ***startAnimacji*** oraz ***animuj***,
- implementacja metod abstrakcyjnych w pozostałych klasach,
- uruchomienie animacji po zakończeniu tworzenia wszystkich obiektów,
- animacja kolejnych klatek animacji,

Zagadnienia: metody abstrakcyjne, dziedziczenie metod,
wywoływanie metod z klasy bazowej,
usypianie wątku – metoda ***sleep*** z klasy ***Thread***.

Krok 7a: Współbieżny wątek do animacji

Kod źródłowy: **RuchomeKulki_07a.java**

Cel: Utworzenie dodatkowej klasy reprezentującej wątek wykonywany współbieżnie i przeniesienie generowania kolejnych klatek animacji do nowego wątku.

- zdefiniowane klasy ***`Animator`*** dziedziczącej po klasie ***`Thread`***,
- przedefiniowanie metody ***`run`*** w klasie ***`Animator`***,
- utworzenie obiektu klasy ***`Animator`*** i uruchomienie wątku metodą ***`start`***,
- uruchomienie animacji wszystkich figur.

Zagadnienia: tworzenie i uruchamianie współbieżnych wątków.

Krok 7b: Współbieżny wątek do animacji (klasa wewnętrzna)

Kod źródłowy: **RuchomeKulki_07b.java**

Cel: Uproszczenie programu 7a przez zdefiniowanie klasy ***Animator*** osadzonej wewnątrz klasy ***Rysunek*** i przeniesienie do obiektu tej klasy procesu animacji rysunku.

- zdefiniowane klasy ***Animator*** dziedziczącej po klasie ***Thread*** jako klasy wewnętrznej osadzonej w klasie ***Ryunek***,
- przedefiniowanie metody ***run*** w klasie ***Animator***,
- utworzenie obiektu klasy ***Animator*** i uruchomienie wątku metodą ***start***,
- uruchomienie animacji wszystkich figur.

Zagadnienia: definiowanie klasy wewnętrznej.

Krok 7c: Współbieżny wątek do animacji (klasa anonimowa)

Kod źródłowy: **RuchomeKulki_07c.java**

Cel: Uproszczenie programu 7b przez utworzenie jednego obiektu klasy anonimowej osadzonej wewnątrz klasy ***Rysunek*** i przeniesienie do obiektu tej klasy procesu animacji rysunku.

- utworzenie za pomocą operatora ***new*** obiektu klasy anonimowej dziedziczącej po klasie ***Thread*** osadzonej w klasie ***Ryunek***,
- przedefiniowanie metody ***run*** w klasie anonimowej,
- uruchomienie wątku metodą ***start***,
- uruchomienie animacji wszystkich figur.

Zagadnienia: definiowanie klasy anonimowej.

Krok 8a: Przyciski „Start” oraz „Dodaj”

Kod źródłowy: **RuchomeKulki_08a.java**

Cel: Dodanie do panelu ***Rysunek*** przycisków „Start” i „Dodaj” oraz zdefiniowanie pomocniczej klasy pełniącej rolę „słuchacza zdarzeń akcji” dla tych przycisków.

- dodanie w klasie ***Rysunek*** metody ***dodajNowaFigura***, która tworzy i dodaje do listy ***listaFigur*** nową figurę o losowych parametrach.
- importowanie klas z pakietu ***java.awt.event***,
- zdefiniowane klasy ***SluchaczZdarzenAkcji*** implementującej interfejs ***ActionListener***,
- zdefiniowanie metody ***actionPerformed*** w klasie ***SluchaczZdarzenAkcji***,
- utworzenie obiektów klasy ***JButton*** reprezentujących dwa przyciski,
- utworzenie obiektu klasy ***SluchaczZdarzenAkcji*** w klasie ***Rysunek***,
- dodanie do przycisków słuchacza zdarzeń i umieszczenie ich w panelu ***Rysunek***,

Zagadnienia: definiowanie klasy implementującej interfejs, dodawanie elementów graficznego interfejsu użytkownika, obsługa „zdarzeń akcji”

Krok 8b: Przyciski „Start” oraz „Dodaj” (klasa wewnętrzna)

Kod źródłowy: **RuchomeKulki_08b.java**

Cel: Uproszczenie programu 8a przez zdefiniowanie klasy ***SluchaczZdarzenAkcji*** jako klasy wewnętrznej osadzonej w klasie ***Rysunek***.

- zdefiniowane klasy ***SluchaczZdarzenAkcji*** implementującej interfejs ***ActionListener*** jako klasy wewnętrznej osadzonej w klasie ***Ryunek***,
- zdefiniowanie metody ***actionPerformed*** w klasie ***SluchaczZdarzenAkcji***,
- utworzenie obiektów klasy ***JButton*** reprezentujących dwa przyciski,
- utworzenie obiektu klasy ***SluchaczZdarzenAkcji*** w klasie ***Rysunek***,
- dodanie do przycisków słuchacza zdarzeń i umieszczenie ich w panelu ***Rysunek***.

Zagadnienia: definiowanie klasy wewnętrznej implementującej interfejs.

Krok 8c: Przyciski „Start” oraz „Dodaj” (klasa anonimowa)

Kod źródłowy: **RuchomeKulki_08c.java**

Cel: Uproszczenie programu 8b przez utworzenie jednego obiektu klasy anonimowej osadzonej wewnątrz klasy ***Rysunek***, który pełni rolę słuchacza zdarzeń akcji.

- w klasie ***Rysunek*** utworzenie za pomocą operatora ***new*** obiektu klasy anonimowej, która implementuje interfejs ***ActionListener***,
- zdefiniowanie metody ***actionPerformed*** w klasie w klasie anonimowej,
- dodanie do przycisków nowego słuchacza zdarzeń.

Zagadnienia: definiowanie klasy anonimowej implementującej interfejs

Krok 8d: Przyciski „Start” oraz „Dodaj” (klasa anonimowa)

Kod źródłowy: **RuchomeKulki_08d.java**

Cel: Uproszczenie programu 8c przez utworzenie dla każdego przycisku obiektu klasy anonimowej implementującego interfejs ***ActionListener*** bezpośrednio w wywołaniu metody ***addActionListener***.

- w wywołaniu metody ***addActionListener*** dla przycisku „Start” należy utworzyć za pomocą operatora ***new*** obiekt klasy anonimowej, która implementuje interfejs ***ActionListener***,
- dla nowej klasy anonimowej należy zdefiniować metodę ***actionPerformed***, która uruchamia animację wszystkich figur,
- w wywołaniu metody ***addActionListener*** dla przycisku „Dodaj” należy utworzyć za pomocą operatora ***new*** obiekt klasy anonimowej, która implementuje interfejs ***ActionListener***,
- dla nowej klasy anonimowej należy zdefiniować metodę ***actionPerformed***, która wywołuje metodę ***dodajNowaFigura***.

Zagadnienia: definiowanie klasy anonimowej implementującej interfejs.

Krok 9: Kolizje figur

Kod źródłowy: **RuchomeKulki_09.java**

Cel: Wykrywanie kolizji dwóch figur i usuwanie mniejszej.

- importowanie klas z pakietu ***java.util.concurrent***,
- zastąpienie kolekcji ***ArrayList<Figura>*** przez kolekcję ***ConcurrentLinkedQueue<Figura>***, która jest bezpieczna dla wątków,
- zdefiniowane w klasie ***Rysunek*** statycznej metody ***sprawdzKolizje***, która wykrywa kolizje dwóch figur,
- rozbudowa metody ***run*** w klasie anonimowej realizującej animację figur,
- przeglądanie kolekcji ***listaFigur*** za pomocą iteratorów i usuwanie kolizyjnych figur za pomocą metody ***remove*** z iteratora.

Zagadnienia: Wykorzystywanie iteratora do przeglądania listy figur, bezpieczeństwo klas dla współbieżnych wątków.

Krok 10: Tworzenie wielu okien graficznych

Kod źródłowy: **RuchomeKulki_10.java**

Cel: Utworzenie kilku niezależnych okien w aplikacji

- utworzenie kilku obiektów klasy ***`RuchomeKulki`*** w metodzie ***`main`***.