Programowanie obiektowe

Język Java – część 2
(przykładowa aplikacja)

Paweł Rogaliński
Instytut Informatyki, Automatyki i Robotyki
Politechniki Wrocławskiej

pawel.rogalinski@pwr.wroc.pl

Zamiast wprowadzenia

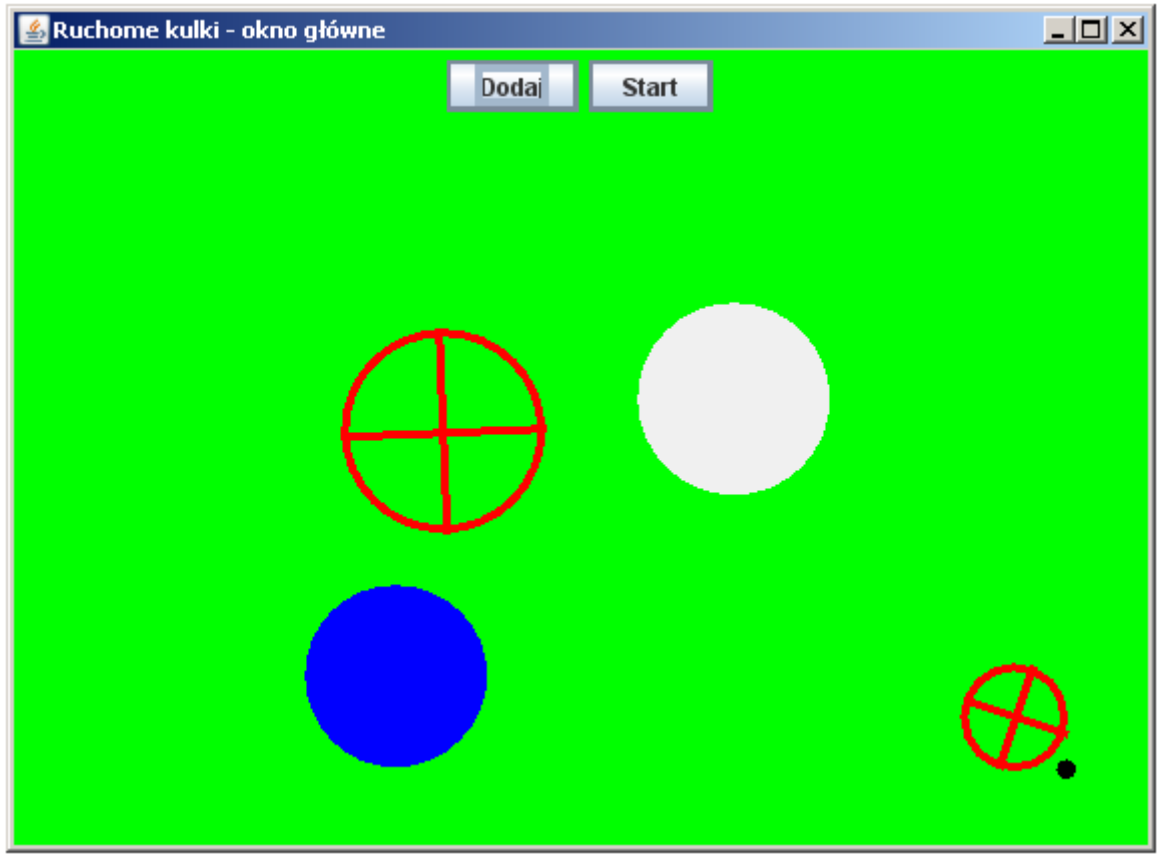
Dla kogo jest dzisiejszy wykład?

- 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

- 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
- 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

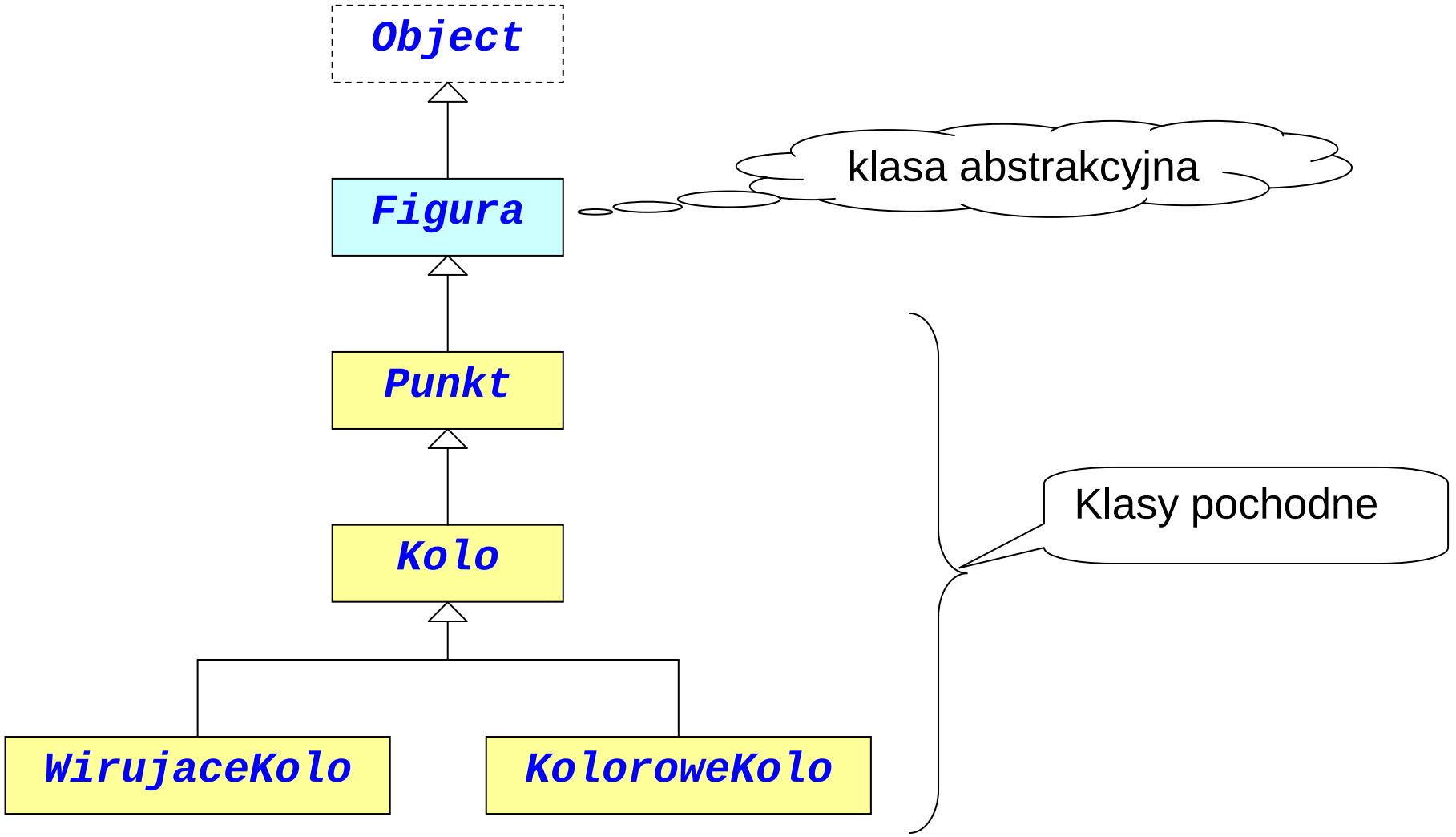
- definicja klasy ***`Figura`***
- utworzenie metody ***`rysuj`***
- utworzenie obiektu reprezentującego figurę geometryczną na rysunku
- wyświetlenie figury w metodzie ***`paintComponent`***

Zagadnienia: wyświetlanie obiektów w panelu rysunku

Krok 4: Klasy reprezentujące różne figury

Kod źródłowy: **RuchomeKulki_04.java**

Cel: Utworzenie hierarchii klas reprezentujących różne figury na rysunku



Krok 4: Klasy reprezentujące różne figury

Kod źródłowy: **RuchomeKulki_04.java**

Cel: Utworzenie hierarchii klas reprezentujących różne figury na rysunku

- definicja abstrakcyjnej klasy ***`Figura`***
- definicja klas pochodnych i implementacja metody abstrakcyjnej rysuj
- utworzenie obiektów reprezentujących różne figury geometryczne
- wyświetlenie wszystkich figur w metodzie ***`paintComponent`***

Zagadnienia: klas 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 różne figury geometryczne
- 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: Animacji 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 7: Współbieżny wątek do animacji

Kod źródłowy: **RuchomeKulki_07.java**

Cel: Przeniesienie generowania kolejnych klatek animacji do dodatkowego wątku

- zdefiniowane dodatkowej 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 7a: Współbieżny wątek do animacji (klasa wewnętrzna)

Kod źródłowy: **RuchomeKulki_07a.java**

Cel: 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 7b: Współbieżny wątek do animacji (klasa anonimowa)

Kod źródłowy: **RuchomeKulki_07b.java**

Cel: 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