kainos®

Wprowadzenie do programowania w Javie

Autor: *Piotr Dubiela*

Interfejsy

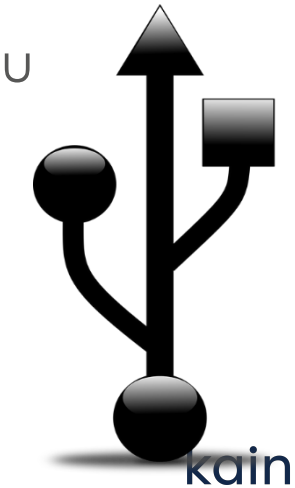
Czym jest interfejs w Javie?

Interfejs (*Interface*) – abstrakcyjny typ, posiadający jedynie operacje i nie posiadający danych.

Interfejsy

Cechy Interfejsu:

- Podobny do klasy abstrakcyjnej, ale nie można zadeklarować konstruktora
- Wszystkie metody są domyślnie publiczne i abstrakcyjne
- Interfejs może rozszerzać jeden lub wiele innych interfejsów
- Klasa może *implementować* wiele interfejsów
- Obiekty klasy implementującej interfejs możemy rzutować na typ Interfejsu
- Interfejs może posiadać jedynie finalne statyczne pola

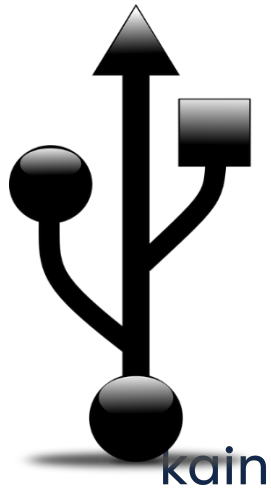


Interfejsy

Interfejs:

Przykładowy interfejs:

```
3  public interface Ruchowy {
4      public void doGory();
5      public abstract void wDol();
6      void wLewo();
7      void wPrawo();
8  }
```



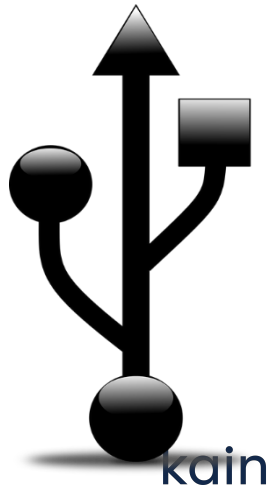
Interfejsy

Interfejs:

Przykładowy interfejs:

```
3  public interface Ruchowy {
4      public void doGory();
5      public abstract void wDol();
6      void wLewo();
7      void wPrawo();
8  }
```

Modyfikatory są bez znaczenia
Każda metoda jest domyślnie public abstract

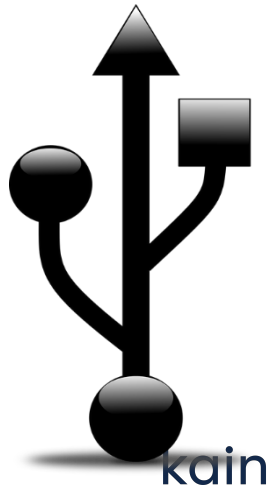


Interfejsy

Interfejs:

```
10  class Gracz implements Ruchowy {
11      int x;
12      int y;
13      int predkosc;
14
15      public Gracz(int predkosc){
16          this.predkosc = predkosc;
17      }
18
19      @Override
20      public void doGory() {
21          x+=predkosc;
22      }
23
```

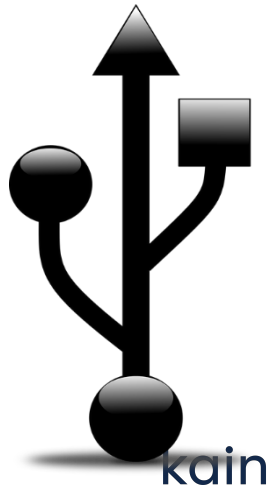
```
24      @Override
25      public void wDol() {
26          y-=predkosc;
27      }
28
29      @Override
30      public void wLewo() {
31          x-=predkosc;
32      }
33
34      @Override
35      public void wPrawo() {
36          x+=predkosc;
37      }
38  }
```



kainos®

Interfejsy

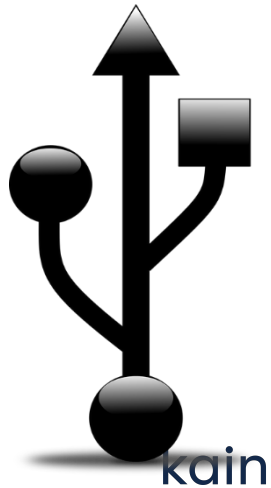
- Nowość w Javie 8:
- Dodano możliwość deklarowania domyślnych metod (*default*), dzięki czemu możemy dodać metodę dla wielu klas implementujących interfejs z domyślnym blokiem kodu
- Dodano metody statyczne, posiadające kod, nie wymagające nadpisywania



Interfejsy

- Metoda default:
- Załóżmy, że posiadamy poniższy interfejs:

```
3  public interface Powiekszalny {
4      int pobierzSzerokosc();
5      int pobierzWysokosc();
6
7      void powieksz0(int wymiar);
8      void poszerz0(int wymiar);
9  }
```



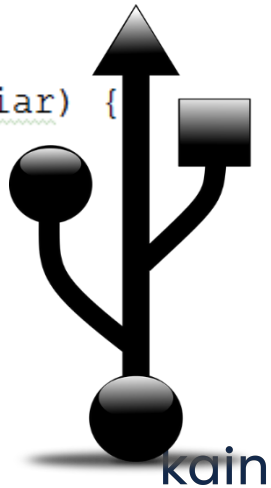
kainos®

Interfejsy

- Metoda default:

```
11  class Okreg implements Powiekszalny{
12      private int szerokosc;
13      private int wysokosc;
14
15      public Okreg(int szerokosc, int wysokosc) {
16          this.szerokosc = szerokosc;
17          this.wysokosc = wysokosc;
18      }
19
20      @Override
21      public int pobierzSzerokosc() {
22          return this.szerokosc;
23      }
```

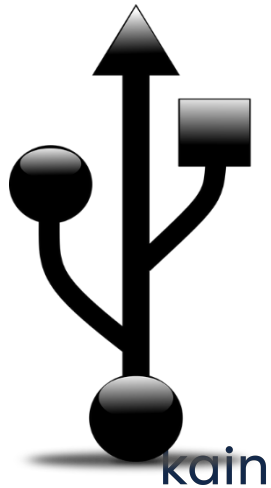
```
25      @Override
26      public int pobierzWysokosc() {
27          return this.wysokosc;
28      }
29
30      @Override
31      public void powiekszO(int wymiar) {
32          this.wysokosc+=wymiar;
33      }
34
35      @Override
36      public void poszerzO(int wymiar) {
37          this.szerokosc+=wymiar;
38      }
39  }
```



kainos®

Interfejsy

- Metoda default:
- Przed JDK 1.8 w celu dodania metody powiekszNRazy należałoby zaktualizować wszystkie klasy implementujące dany interfejs



kainos®

Interfejsy

- Metoda default:
- Od JDK 1.8 wystarczy dodanie metody default:

```
3   public interface Powiekszalny {
4       int pobierzSzerokosc();
5       int pobierzWysokosc();
6
7       void powieksz0(int wymiar);
8       void poszerz0(int wymiar);
9
10      default void powiekszNRazy(int n){
11          int aktualnaWysokosc = pobierzWysokosc();
12          for (int i=1; i<n; i++){
13              powieksz0(aktualnaWysokosc);
14          }
15      }
16
17      default void poszerzNRazy(int n){
18          int aktualnaSzerokosc = pobierzSzerokosc();
19          for (int i=1; i<n; i++){
20              poszerz0(aktualnaSzerokosc);
21          }
22      }
23  }
```

kainos®

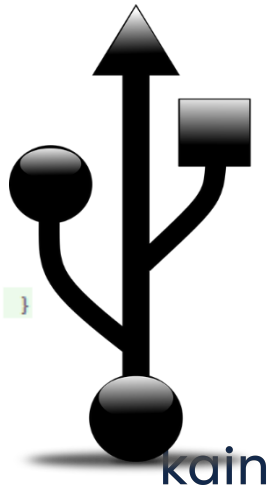
Interfejsy

Comparable:

Interfejs pozwalający na sortowanie obiektów dowolnego typu

Metoda compareTo() powinna zwrócić 0 jeśli obiekty są sobie równe

```
5  public class Czlowiek implements Comparable{
6      String imie;
7      String nazwisko;
8      Plec plec;
9
10     public Czlowiek(String imie, String nazwisko, Plec plec) {
11         this.imie = imie;
12         this.nazwisko = nazwisko;
13         this.plec = plec;
14     }
15
16     public Plec pobierzPlec() { return plec; }
19
20     @Override
21     public String toString() { return String.format("%s : %s %s", plec, imie, nazwisko); }
24
25     @Override
26     public int compareTo(Object o) {
27         Czlowiek that = (Czlowiek)o;
28         return nazwisko.compareTo(that.nazwisko);
29     }
```

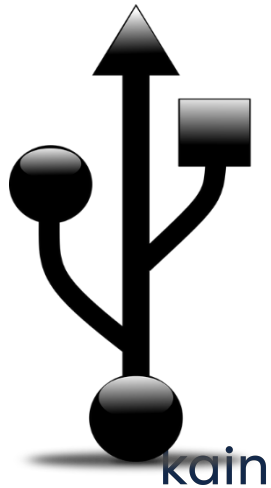


kainos®

Interfejsy

Comparable:

```
31      public static void main(String[] args) {
32          Czlowiek adam = new Czlowiek( imie: "Adam",  nazwisko: "Kowalski", Plec.MEZCZYZNA);
33          Czlowiek malgorzata = new Czlowiek( imie: "Agnieszka",  nazwisko: "Adamowicz", Plec.KOBIETA);
34          Czlowiek panSamolot = new Czlowiek( imie: "Jerzy",  nazwisko: "Rokicki", Plec.SAMOLOT);
35
36          Czlowiek[] ludki = new Czlowiek[]{adam, malgorzata, panSamolot};
37          wyswietlLudkow(ludki);
38          System.out.println("Sortowanie ");
39          Arrays.sort(ludki);
40          wyswietlLudkow(ludki);
41      }
42
43      private static void wyswietlLudkow(Czlowiek[] ludki) {
44          for(Czlowiek ludek:ludki){
45              System.out.println(ludek);
46          }
47      }
```



kainos®

Czas w Javie – LocalDate, LocalTime, LocalDateTime

Czas w Javie możemy wyrazić za pomocą specjalnych klas:

LocalDate – zawiera informacje o dacie w formacie rok-miesiąc-dzień

LocalTime – zawiera informacje o czasie w formacie godzina:minuta:sekunda

LocalDateTime – połączenie powyższych klas, zawiera informacje o dacie i czasie w formatach podanych powyżej

LocalDate

```
LocalDate now = LocalDate.now();

LocalDate date1 = LocalDate.of(2020, 5, 4);
LocalDate date2 = LocalDate.parse("2020-5-4");

int year = date1.getYear(); // 2020
int month = date1.getMonthValue(); // 5
int dayOfMonth = date1.getDayOfMonth(); // 4
int dayOfYear = date1.getDayOfYear(); // 125

int lengthOfYear = date1.lengthOfYear(); // 366
int lengthOfMonth = date1.lengthOfMonth(); // 31

LocalDate tomorrow = date1.plusDays(1);
LocalDate yesterday = date1.minusDays(1);
LocalDate inTwoYears = date1.plusYears(2);
LocalDate in2016 = date1.withYear(2016);
```

LocalTime

```
LocalTime now = LocalTime.now();


LocalTime.of(11, 45);        // 11:45
LocalTime.of(11, 45, 30);    // 11:45:30
LocalTime.parse("11:45:30"); // 11:45:30

time.getHour();   // 11
time.getMinute(); // 45
time.getSecond(); // 30
time.getNano();   // 0
```

```
LocalTime time1 = time.plusHours(5); // 16:45:30
LocalTime time2 = time.plusHours(22); // 09:45:30
LocalTime time3 = time.minusMinutes(10); // 11:35:30
LocalTime time4 = time.minusSeconds(30); // 11:45


LocalTime time1 = time.withHour(23); // 23:45:30
LocalTime time2 = time.withMinute(50); // 11:50:30
LocalTime time3 = time.withSecond(0); // 11:45
```

LocalDateTime

```
LocalDateTime now = LocalDateTime.now();


LocalDateTime dt1 = LocalDateTime.of(2017, 11, 25, 22, 30);  // 25 November 2017, 22:30
LocalDateTime dt2 = LocalDateTime.parse("2017-11-25T22:30");


LocalDate date = LocalDate.of(2017, 11, 25); // 2017-11-25
LocalTime time = LocalTime.of(21, 30); // 21:30

LocalDateTime dateTime = LocalDateTime.of(date, time); // 2017-11-25T21:30


LocalDate date = LocalDate.of(2017, 11, 25); // 2017-11-25
LocalTime time = LocalTime.of(21, 30); // 21:30

LocalDateTime dateTime1 = date.atTime(time); // 2017-11-25T21:30
LocalDateTime dateTime2 = time.atDate(date); // 2017-11-25T21:30
```

LocalDateTime cd.

- int month = dateTime.getMonthValue(); // 11
  int day = dateTime.getDayOfMonth(); // 25
  int hour = dateTime.getHour(); // 22
  int minute = dateTime.getMinute(); // 30

- LocalDate dateOf = dateTime.toLocalDate(); // 2017-11-25
  LocalTime timeOf = dateTime.toLocalTime(); // 22:30

Formatowanie dat - DateTimeFormatter

DateTimeFormatter pomoże nam w sformatowaniu daty.

```
LocalDate date = LocalDate.of(2020, 5, 4);
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
String formattedDate = date.format(formatter); // 04-05-2020
```

Wszystkie możliwości patternów:

https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html

Klasa File

Przykładowe metody klasy File:

String **getPath() – zwraca ścieżkę do pliku**

String **getName() – zwraca nazwę pliku**

boolean **isDirectory() – sprawdza, czy plik jest katalogiem**

boolean **isFile() – sprawdza, czy plik jest plikiem**

boolean **exists() – sprawdza, czy plik istnieje**

String **getParent() – zwraca ścieżkę do katalogu nadrzędnego**

File **getParentFile() – zwraca obiekt File katalogu nadrzędnego**

String[] **files() – zwraca listę ścieżek wszystkich plików w katalogu**

File[] **listFiles() – zwraca obiekty File wszystkich plików w katalogu**

Działania na plikach

W Javie do działań na plikach używamy klasy „File”.

File file = new File(„/sciezka/do/pliku”);

Należy pamiętać, że tak stworzony obiekt jest jedynie wirtualnym odnośnikiem do pliku – nie tworzy on fizycznie pliku w żaden sposób.

Tworzenie, przenoszenie i usuwanie plików

boolean createNewFile() – tworzy nowy plik

boolean mkdir – tworzy katalog

boolean mkdirs – tworzy wszystkie potrzebne katalogi

boolean delete() – usuwa plik / katalog

boolean renameTo() – zmienia nazwę / przenosi plik

Ścieżki w plikach - absolutne

Przykłady ścieżek absolutnych:

```
File fileOnUnix = new File("/home/username/Documents");    // UNIX
File fileOnWin = new File("D:\\Courses\\java-course.pdf");  // Windows
```

Ścieżki absolutne w systemie Windows zaczynają się od litery partycji. Ścieżki absolutne w systemie UNIXowym zaczynają się od “/”.

Należy zwrócić uwagę, że w systemie UNIXowym pliki i katalogi w ścieżce oddzielamy znakiem slash ”/”, zaś w Windows backslash “\”. W Javie backslash jest znakiem specjalnym, dlatego musimy użyć podwójnego “\\”

Ścieżki w plikach - relatywne

Przykłady ścieżek relatywnych:

```
File fileOnUnix = new File("./images/picture.jpg"); // Unix
File fileOnWin = new File("./images/picture.jpg"); // Windows
```

Ścieżki relatywne to takie, które położone są względem katalogu w którym się znajdujemy.

Znak kropki “.” oznacza obecny katalog.

Znak podwójnej kropki “..” oznaczają katalog rodzica (jeden wyżej niż ten w którym się znajdujemy).

W przypadku ścieżek relatywnych możemy używać “/” niezależnie od system operacyjnego.

Wczytywanie plików

```
File file = new File(“sciezka/do/pliku”);
Scanner scanner = new Scanner(file);

while (scanner.hasNext()) {
    System.out.print(scanner.nextLine());
}
```

albo…

```
String fileString = Files.readAllBytes(Paths.get(”sciezka/do/pliku”));
```

Uwaga na wyjątek.

Zapisywanie do pliku – klasa FileWriter

FileWriter(String fileName);

FileWriter(String fileName, boolean append);

FileWriter(File file);

FileWriter(File file, boolean append);

Opcja append określa czy tekst ma być dopisywany na końcu pliku zamiast nadpisywania całej treści pliku.

```
writer.write("Hello World);

writer.close();
```

Zapisywanie do pliku – klasa PrintWriter

PrintWriter(String fileName);

PrintWriter(File file);

printWriter.print(123);

printWriter.println(456);
printWriter.printf("You have %d %s", 400, "gold coins");

try with resources

W celu uniknięcia wycieku pamięci powinniśmy zawsze zamykać takie obiekty jak FileWriter, PrintWriter czy Scanner. Zamiast tego możemy użyć struktury „try with resources" która zrobi to za nas po zakończeniu operacji:

```
try (Scanner scanner = new Scanner(file)) {
    while (scanner.hasNext()) {
        System.out.print(scanner.nextLine() + " ");
    }
}
```

```
try (FileWriter writer = new FileWriter(file)) {
    writer.write("Hello, World");
}
```

Bibliografia

1. https://pixabay.com/pl/młotek-narzędzia-metalowe-celuj-w-33617/ (dostęp 28.12.2017)
2. *https://pixabay.com/pl/bunnies-królików-przedsiębiorstwo-151390/ (dostęp 28.12.2017)*
3. *http://www.baeldung.com/java-varargs (dostęp 28.12.2017)*
4. https://pixabay.com/pl/niebezpieczeństwo-panelu-uwagi-3061159/ (dostęp 29.12.2017)
5. https://docs.oracle.com/javase/tutorial/essential/exceptions/catchOrDeclare.html (dostęp 29.12.2017)
6. https://pixabay.com/pl/pokemon-pokeball-pokemon-idź-1536849/ (dostęp 30.12.2017)
7. https://pixabay.com/pl/banderą-formuła-chequered-speedway-42581/ (dostęp 30.12.2017)
8. https://pl.wikipedia.org/wiki/Wyrażenie_regularne (dostęp 30.12.2017)
9. https://pl.wikipedia.org/wiki/Interfejs_(programowanie_obiektowe) (dostęp 30.12.2017)
10. https://pixabay.com/pl/symbol-usb-komputery-symbol-1906474/ (dostęp 30.12.2017)
11. https://pixabay.com/pl/człowiek-głowy-twarz-awatar-157699/ (dostęp 31.12.2017)
12. https://pl.wikipedia.org/wiki/Programowanie_uogólnione (dostęp 31.12.2017)
13. https://pixabay.com/pl/szafki-na-ubrania-półka-szafki-575373/ (02.01.2018)
14. https://docs.oracle.com/javase/tutorial/collections/interfaces/list.html (dostęp 02.01.2018)
15. https://www.javatpoint.com/difference-between-arraylist-and-linkedlist (dostęp 02.01.2018)

kainos®