**I/O – obsługa plików w Javie**

W Javie posiadamy możliwość zapisu i odczytu do pliku. Możemy obsługiwać zarówno pliki binarne jak i tekstowe.

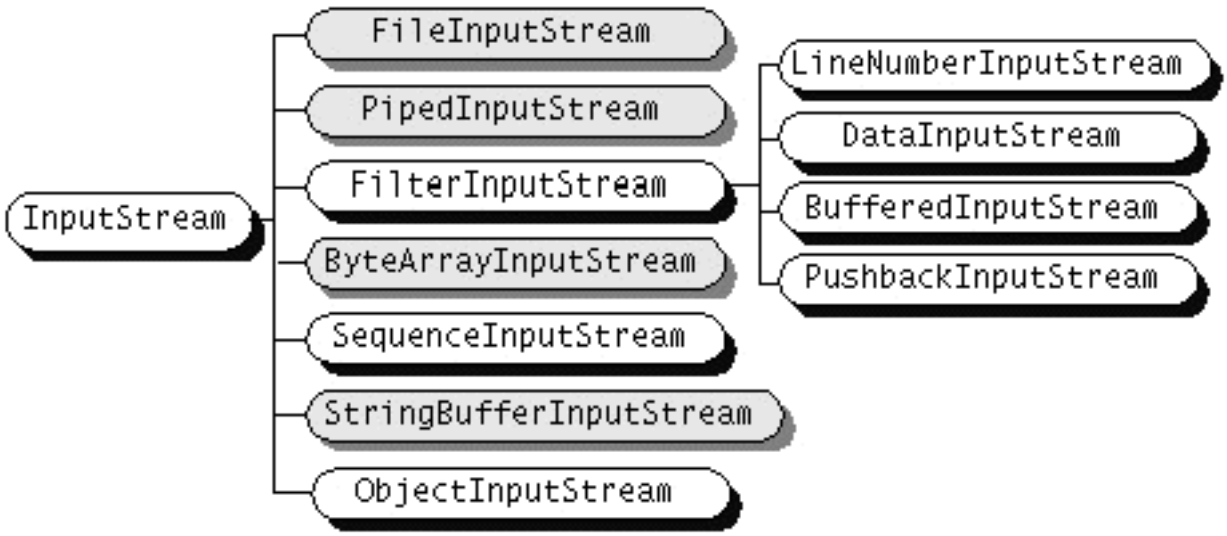
Z tematem obsługi plików wiąże się Serializacja. Jest to proces zapisu i odczytu obiektu przykładowo do pliku w formie binarnej. Serializować możemy obiekty klas implementujących interfejs Serializable.

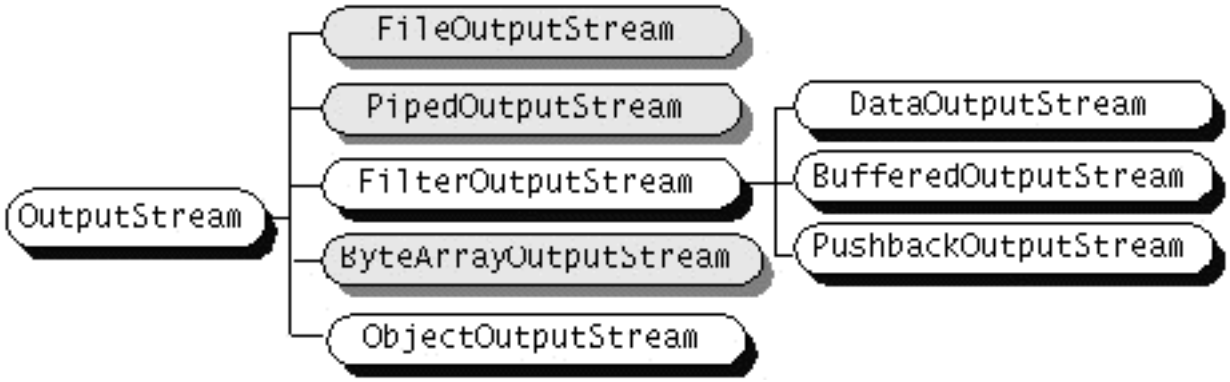
Obsługa I/O w Javie opiera się na strumieniach danych. Rozróżniamy dwa typy strumieni:

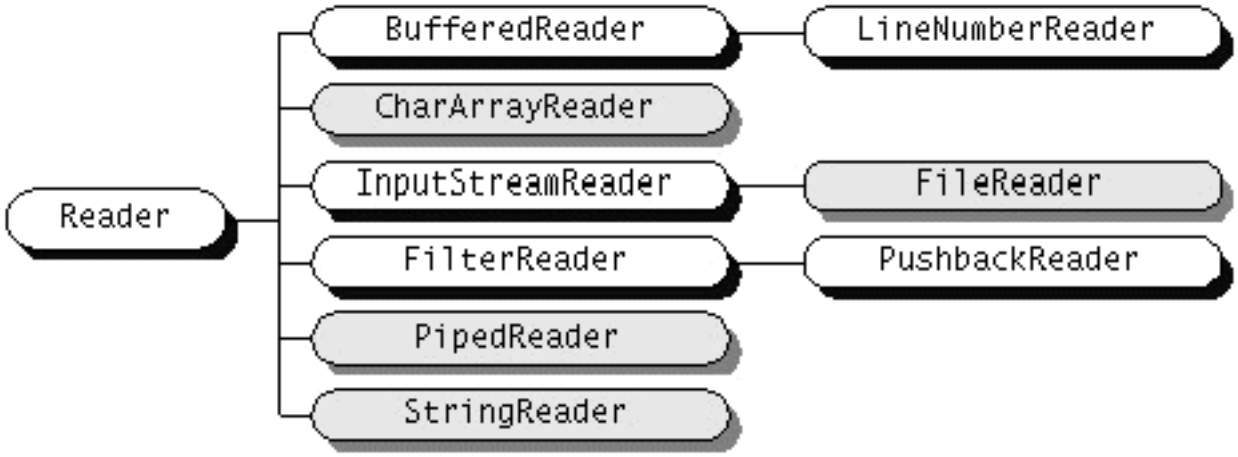
- bajtowe – dane w nieczytelnej dla nas formie bajtów (InputStream, OutputStream)

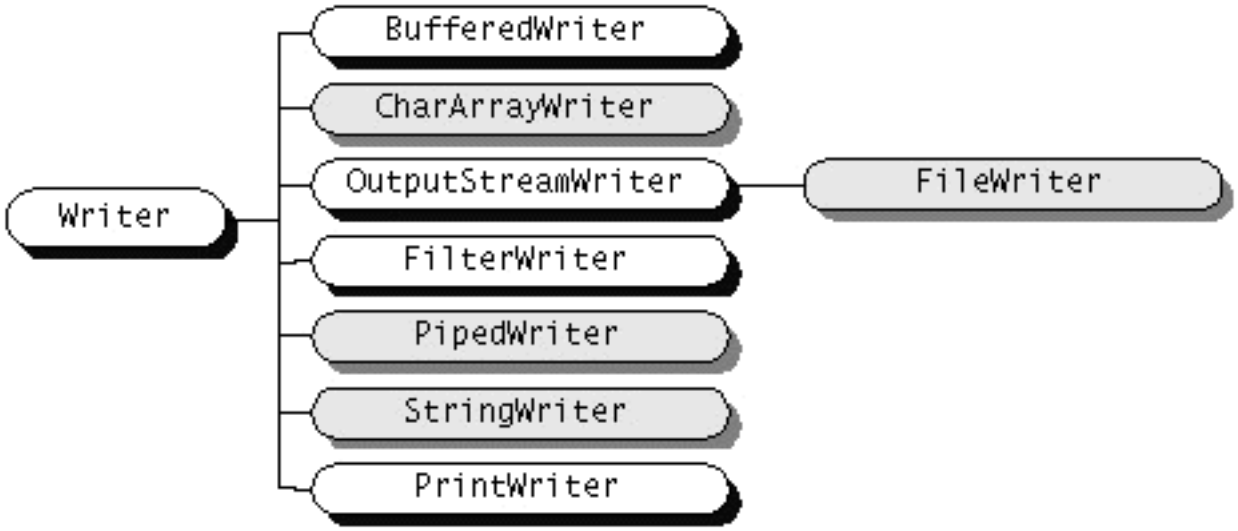
- znakowe – dane w czytelnej formie ciągów znakowych (Reader, Writer).

Zależnie od zastosowania możemy je podzielić na:






Ważną podgrupę strumieni stanowią strumienie buforowane pozwalające na użycie bufora do zapisu i odczytu pliku – pozwala to odczytywać plik po kawałku oraz przyspieszyć zapis do pliku.

**Odczyt pliku tekstowego**

Do odczytu pliku tekstowego możemy wykorzystać obiekt klasy BufferedReader. Daje ona możliwość odczytania pliku w sposób buforowany (tj. po kawałku). Nowy obiekt klasy BufferedReader możemy stworzyć za pomocą metody statycznej Files.newBufferedReader. Do utworzenia readera będziemy potrzebować obiektu ścieżki (Path) do pliku oraz Charset – kodowanie znaków – dobrowolnie.

Klasa BufferedReader ma dwie interesujące nas metody :

- readLine – odczyt kolejnej linii pliku, jeśli wynik == null to oznacza koniec pliku

- lines – zwraca strumień linii

Plik możemy odczytać jeden raz, odczytanie linii powoduje przejście kursora do kolejnej. Jeśli chcemy odczytać plik ponownie musimy go ponownie otworzyć.

**Odczyt pliku binarnego**

Do odczytu małych plików binarnych możemy wykorzystać metodę klasy Files – readAllBytes. Przekazujemy ścieżkę do pliku. Metoda zwraca nam tablicę bajtów. Nie wykorzystujcie tej metody do odczytu dużych plików.

**Zapis pliku tekstowego**

Do zapisu do pliku tekstowego możemy wykorzystać obiekt klasy BufferedWriter. Nowy obiekt klasy BufferedWriter możemy stworzyć za pomocą metody statycznej Files.newBufferedWriter. Do stworzenia writera będziemy potrzebować obiektu ścieżki do pliku oraz Charset (dobrowolne). Potrzebne nam będzie także określenie opcji otwarcia pliku. Lista dostępnych:

https://docs.oracle.com/javase/7/docs/api/java/nio/file/StandardOpenOption.html

Writer ma ważną dla nas metodę:

- write – pozwalająca na zapis do pliku

Aby przejść do nowej linii należy zakończyć tekst znakiem nowej linii lub wywołać metodę newLine z klasy BufferedWriter

**Zapis pliku binarnego**

Aby zapisać tablicę bajtów w pliku możemy skorzystać z metody write z klasy Files. Wystarczy że przekażemy ścieżkę do pliku oraz tablicę bajtów oraz opcję otwarcia pliku.

**Try with resources**

W Javie 7 do klasycznego bloku try-catch-finally dołączył blok try with resources. Blok ten zwalnia nas z konieczności zamknięcia strumienia danych, który został otworzony wewnątrz bloku. Nie zależnie od sukcesu operacji w bloku try strumień zawsze zostanie zamknięty.

```
try (tworzenie obiektu strumienia) {

… kod mogący rzucać wyjątek

} catch (IOException ex) {

}
```

Nie chcąc używać tej konstrukcji, musielibyśmy obiekt strumienia stworzyć przez blokiem try-catch a następnie wywołać na nim metodę close w bloku finally.

```
Path sciezka = Paths.get(plik);

try (BufferedReader reader = Files.newBufferedReader(sciezka)) {
    return reader.lines()
            .map(linia -> linia.split(";"))
            .map(DruzynaPilkarska::new)
            .collect(Collectors.toList());

} catch (IOException ex) {
    ex.printStackTrace();
}
```