1. Pliki i Katalogi

Java jest wyposażona jest w narzędzie biblioteczne do obsługi plików. Jest nim klasa File, która odnosi się nie do samego pliku lecz do ścieżki pliku. Klasa bowiem reprezentuje nazwę pliku lub nazwę zbioru plików w katalogu.

Osobną klasą jest RandomAccessFile, którą używa się do obsługi plików zawierających rekordy o znanym rozmiarze, tak aby można było między nimi przemieszczać się za pomocą metody seek(), czytać je lub zmieniać. Nie ma konieczności aby miały one ten sam rozmiar, wystarczające jest aby można było określić ich wielkość i dokładne rozmieszczenie w pliku.

Klasa File

java.lang.**Object**
\--java.io.**File**

```
File path = new File("."); //wskazanie na aktualny katalog
```

Klasa RandomAccessFile

java.lang.**Object**
\--java.io.**RandomAccessFile**

2. Strumienie

Biblioteka strumieni wejścia-wyjścia Javy spełnia podstawowe wymagania: pozwala na odczytywanie i zapisywanie danych na konsolę, do pliku, bloku pamięci, a także przez Internet. Przez dziedziczenie można tworzyć nowe typy obiektów akceptowanych przez strumień, przedefiniowując metodę toString(), która jest wywołana automatycznie, kiedy przekazuje się obiekt metodzie oczekującej jako argumentu ciągu znaków. W Javie hierarchia strumieni oparta jest na czterech klasach InputStream, OutputStream, Reader oraz Writer. InputStream oraz Reader reprezentują strumienie danych wejściowych natomiast dwa pozostałe są strumieniami danych wyjściowych.

Podklasy InputStream oraz OutputStream	Podkalsy Reader oraz Writer	Opis
FileInputStream FileOutpuStream	FileReader FileWriter	Pozwala odczytywać i zapisywać pliki dyskowe. Jako parametr konstruktora należy przekazać nazwę pliku

		dyskowego lub wskazujący go obiekt File.
ByteArrayInput ByteArrayOutput	CharArrayRader CharArrayWriter	Bufor w pamięci oparty na tablict odpowiednio bajtów lub znaków
StringBufferInputStream (nie ma odpowiednika zapisu)	StringReader StringWriter	Bufor w pamięci oparty na o String (implementacja posługuje się obiektem StringBuffer). Tworząc obiekt wejściowy należy przekazać konstruktorowi String na którym obiekt ma być oparty.
PipiedInputStream PipedOutputStream	PippedReader PippedWriter	Łącze do komunikacji pomiędzy procesami. Między strumieniami zostanie ustawione łącze dzięki któremu możliwe jest przesyłanie danych od strumienia wyjściowego do wejściowego.
SequenceInputStream		Konwertuje dwa lub więcej obiektów InputStream do pojedynczego InputStream
FilterInputStream FilterOutputStream	FilterReader FilterWriter (klasa abstrakcyjna nie posiada podklas)	Abstrakcyjne klasy służące jako interfejs dla klas „dekoratorów” dostarczających dodatkowe możliwości innym klasom InputStream oraz OutputStream.
BufferedInputStream BufferedOutputStream	BufferedReader(posiada metodą readLine()) BufferedWriter	Klasy używane aby zapobiec fizycznemu odczytowi/zapisowi za każdym żądaniem kolejnych danych. Nie dostarcza

		własnego interfejsu, jedynie wymusza użycie bufora. Należy dołączyć interfejs.	3

3. Serializacja

Istniejący w Javie mechanizm *serializacji obiektów* pozwala na zmianę dowolnego obiektu, który implementuje interfejs *Serializable*, na sekwencję bajtów w sposób który umożliwia jego wierne odtworzenie w późniejszym czasie. Mechanizm ten działa również w sieci, co oznacza że automatycznie kompensuje różnice pomiędzy systemami operacyjnymi. Co za tam idzie obiekt stworzony np. na systemie operacyjnym Windows po przesłaniu przez sieć do maszyny uniksowej zostanie poprawnie odtworzony. Mechanizm ten zwalnia nas z konieczności zadbania o reprezentację danych w różnych systemach, uporządkowanie bajtów czy też inne szczegóły. Serializacja umożliwia zaimplementowanie *lekkiej trwałości*, co oznacza że czas życia obiektu nie ogranicza się jedynie do czasu życia programu. Dzięki zapisaniu zserializowanego obiektu na dysku pozwala na odtworzeniu go przy ponownym uruchomieniu programu, co powoduje uzyskanie efektu trwałości. Dzięki serializacji dane można przechowywać nie tylko w postaci binarnej ale również znakowaj (np. XML).

Metody interfejsu *Serializable:*

- writeObject() - wywoływane przez ObjectOutputStream(OutputStrem)
- readObject() - wywoływane przez ObjectInputStream(InputStrem)

Pola **transient** oraz **static** nie są serializowane

4. Externalizable

W przypadku gdy mechanizm *Serializacji* okazuje się niewystarczający. Może to nastąpić w przypadku gdy dla bezpieczeństwa nie należy serializować pewnych części obiektu, bądź też nie ma sensu serializować pod obiektów, jeżeli i tak będą musiały zostać stworzone od nowa po utworzeniu obiektu. W takim przypadku procesem serializacji można sterować, implementując interfejs *Externalizable* zamiast *Serializable*. Interfejs *Externalizable* rozszerza ten pierwszy o dwie dodatkowe metody, wywoływane

automatycznie podczas procesu serializacji i deserializacji obiektu, tak aby możliwe było wykonanie specjalnych operacji.

Metody interfejsu *Externalizable:*

- writeExternal() - wywoływane przez ObjectOutputStream(OutputStrem)
- readExternal() - wywoływane przez ObjectInputStream(InputStrem)

Zadanie:

- napisać aplikację wykonującą zapis i odczyt danych dowolnego typu(typy proste – liczby losowe typu double, łańcuchy znaków – wczytywanie tekstu z klawiatury)
- wykonać aplikację zapisującą i odczytującą obiekty zawierające pola nieserializowane
- zastosować buforowanie