1 Pakiety

Pakiety w javie jest to mechanizm grupowania klas w większe jednostki (przestrzenie nazw). W pakiety grupuje się klasy realizujące wspólnie jedno zadanie np. klasy tworzące GUI, lub mające inne cechy wspólne które sprawiają że sensownie jest myślec o nich jako o pewnym zbiorze, np klasy implementujące różne algorytmy sortowania, kontrolki GUI, serwlety itp.

Aby umieścić klasę w pakiecie należy w pliku .java na samym początku umieścić instrukcję deklaracji pakietu `package` np.:

```
package mojpakiet;

public class MojaKlasa {
    int a;
}
```

Instrukcja package powoduje że wszystkie klasy zadeklarowane w damym pliku trafiają do odpowiedniego pakietu. Teraz aby użyć klasy MojaKlasa w innym pliku można albo ja zaimportować pisząc na początku pliku import mojpakiet.MojaKlasa; albo odwołać się do niej przez jej *nazwę kwalifikowaną*: mojpakiet.MojaKlasa o = new mojpakiet.MojaKlasa() Jedynym wyjątkiem od tej reguły są klasy znajdujące się w pakiecie java.lang (Object, Class, String, Integer, Thread, ...) będące de facto elementem języka, ich nie trzeba importować ani używać nazw kwalfikowanych.

Klasy znajdujące się w tym samym pakiecie są ze sobą dodatkowo związane, mianowicie mają one dostęp do swoich pól i metod o dostępie pakietowym (zadeklarownych bez słówek public,protected lub private). W przykładzie wyżej inne klasy z pakietu mojpakiet mają dostęp do pola a tak jak by było ono publiczne, dla klas z poza pakietu mojpakiet jest ono prywatne.

Jak wiadomo w środowisku javy pliki z kodem źródłowym muszą mieć ściśle określone nazwy, zgadzające się z nazwami publicznych klas w nich zadeklarowanych. Okazuje się miejsce gdzie one się znajdują na dysku też nie jest zupełnie dowolne. Katalogi w których znajdują się pliki źródłowe (i gdzie muszą znaleźć się pliki .class) są określone przez pakiety do których one należą np. ścieżka do pliku z klasą moja klasa musi wyglądać tak: mojpakiet/MojaKlasa.java, plik .class trafi tu: mojpakiet/MojaKlasa.class. Katalog w którym znajduje się pakiet jest już dowolny.

Każda klasa musi znajdować się w jakimś pakiecie, jeśli w pliku nie było deklaracji pakietu to wszystkie klasy trafiają do tzw. pakietu domyślnego, odpowiednie pliki znajdują się wtedy w głównym katalogu. Dlatego polecenia takie jak javac HelloWorld.java i później java HelloWorld działają. Korzystanie z domyślnego pakietu jest złą praktyką.

Nazwy pakietów mają strukturę hierarchiczną w której poszczególne elementy nazwy są rozdzielone kropką, na przykład klasa MojaKlasa może się znajdować w pakiecie pl.edu.pw.java wtedy ścieżka do pliku .java będzie taka: pl/pw/edu/java/MojaKlasa.java (elementami nazwy pakietu nie moga być słowa kluczowe javy, dlatego nie może tam być członu "if"). Standardową praktyką nazwania pakietów jest użycie odwróconej nazwy domeny organizacji do której należy dany kod np. biblioteka GWT tworzona przez Google znajduje się w pakietach zaczynających się od com.google.gwt., biblioteka Spring: org.springframework. itd. Ponieważ nazwy domenowe są założenia unikalne, taka metoda nazywania klas zapobiega konfliktom. Jedynym wyjątkiem od tej reguły jest stanardowa biblioteka javy która znajduję się w pakietach java. i javax. .

2 CLASSPATH

Java jest środowiskiem w którym klasy składające się na program są ładowane *dynamicznie* oraz *leniwie* (ang. lazy). Oznacza to że kod oraz dane składające się na klasę są ładowane do pamięci przez maszynę wirtualną dopiero w momencie pierwszego użycia jej użycia. Jednak aby maszyna wirtualna, a konkretnie tzw. *ClassLoader* mógł załadować klasę musi wiedzieć gdzie ma szukać konkretnego pliku .class. Klasy składające się na bibliotekę standardową znajdują się w dobrze określonym, zawsze tym samym miejscu, nie ma więc kłopotu z ich odnaleźeniem, jednak nasze klasy oraz zewnętrzne biblioteki mogą znajdować się gdziekolwiek (istnieje nawet możliwośc ładowania klas przez sieć). CLASSPATH jest parametrem

przekazywanym maszynie wirtualnej zawierającym listę lokalizacji (pod Linuksem rozdzielonych dwukropkiem, pod Windowsem średnikiem) gdzie znajdują się klasy, składające się na program. Parametr ten może być przekazywany w jako parametr -cp lub -classpath w wierszu polecenia, lub przez zmienną środowiskową CLASSPATH. Lokalizacje te mogą stanowić katalogi w systemie plików lub archiwa JAR. Na przykład jeśli chcemy aby maszyna wirtualna zaczęła pracę od metody main znajdującej się w klasie pl.test.Test znajdującej się w katalogu /home/user/classes powinniśmy wydać polecenie:

```
java -classpath /home/user/classes pl.test.Test
```

Pełna ścieżka do pliku Test.class powinna być taka: /home/user/classes/pl/test/Test.class. Aktualny katalog domyslnie jest elementem classpath, więc jeśli poprzednie polecenie wydalibyśmy znajdując się w katalogu /home/user/classes to parametr -classpath można by pominąć.

3 Archiwa JAR

Ponieważ biblioteki i programy pisane w javie mogą składać się nawet z tysięcy klas, rozprowadzanie ich jako zbioru plików .class w odpowiedniej strukturze katalogów jest niepraktyczne. Tu z pomoca przychodzą pliki .jar (Java ARchive). Plik .jar to archiwum w formacie zip zawierające pliki .class, potrzebne im zasoby oraz, czasami, kod źródłowy. Jeśli chcemy skorzystać z klas umieszczonych w pliku .jar musimy umieścić go w classpath np.

```
java -cp JakasBiblioteka.jar pl.test.MainClass
```

W ten sam sposób można dodać też zawartość zwykłego archiwum zip.

Plik .jar oprócz struktury katalogów odpowiadającej pakietom powinien też zawierać katalog META-INF a w nim plik MANIFEST.MF . Jest to plik tekstowy, w formacie odpowiadającym klasie java.util.Properties, zawierający informacje o bibliotece. W szczególności może on zawierać atrybut Main-Class określający którą klasę uruchomić przy wywołanu java -jar. Katalog META-INF może zawierać też inne pliki konfiguracyjne, zależne od środowiska w którym dany jar ma być używany, korzystają z tego np. komponenty EJB, oraz różnego rodzaju mechanizmy wtyczek.