Laboratorium 4

1. Prezentacja: JUnit.
   Po wygłoszeniu referat do wysłania mailem (format PDF + edytowalny) na adres: skrzynia@agh.edu.pl, tytuł maila:
   [PZ1] Referat JUnit– grupa *$grupa$* – nazwiska autorów oddzielone przecinkiem
   przy czym *$grupa$* przyjmuje wartości: pn. 18:30, śr. 8, śr. 9:45, śr 11:30
   np.:
   [PZ1] Referat JUnit – grupa pn. 18:30 – Nowak, Kowalski
   Referat należy też KONIECZNIE udostępnić kolegom/koleżankom z grupy.
   Zrealizuj zaległe punkty z lab. 3 jeśli tego nie zrobiłeś.
2. Sprawdź czy po zbudowaniu i uruchomieniu paczki jar z programem z lab. 3 (po dodaniu zależności od apache commons) jest zgłaszany wyjątek NoClassDefFoundException.
3. Do programu z lab. 3 dodaj plugin Maven Asscembly plugin poprzez dodanie do pliku pom.xml w sekcji `<build><plugins>` nowej sekcji:
```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>2.6</version>
  <configuration>
    <appendAssemblyId>false</appendAssemblyId>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
    <archive>
      <manifest>
        <mainClass>pakiet.NazwaKlasyZMain</mainClass>
      </manifest>
    </archive>
  </configuration>
</plugin>
```
4. Zbuduj z lini poleceń A następnie wpisać w konsolę
   `mvn clean compile assembly:assembly`
   (w nowszych wersjach pluginu: `assembly:single`)
5. Uruchom program z linii poleceń przez java -jar.

Uwaga: jeśli korzystasz z dostarczonego przykładu multi-module.zip to zwróć uwagę, że dostarczony przykład buduje plik jar zawierający wszystkie zależności (main-1.0-jar-with-dependencies.jar w module main)

6. W szablonie zadania 1, które stworzyłeś podczas lab. 3 dodaj do modułu utils klasę MyMap implementującą inteface Map (z metodami jak niżej):

```
public interface Map<K, V> {
   /**
    * Dodanie elementu do mapy pod podanym kluczem.
      Jeśli podany klucz istnieje to metoda powinna podmienić wartość.
    * @param key klucz (nie null)
    * @param value wartość kryjącą się pod kluczem (nie null)
    * @return true jeśli się uda dodać element, false jeśli nie
    */
   boolean put(K key, V value);

   /**
    * Usunięcie podanego klucza oraz wartości, która jest przechowywana pod tym
kluczem.
    * @param key klucz do usunięcia
    * @return true jeśli uda się usunać klucz, false w przeciwnym przypadku
```

```
     */
    boolean remove(K key);

    /**
     * Zwraca wartość pod podanym kluczem lub null jeśli nie ma podanego klucza.
     * @param key klucz (nie ull)
     * @return wartość pod kluczem lub null jeśli nie ma wartości dla podanego klucza
     */
    V get(K key);

    /**
     * Zwraca listę kluczy
     * @return java.util.List lista kluczy
     */
    List<K> keys();

    /**
     * Sprawdza czy podany klucz istnieje w mapie.
     * @param key wartość klucza do sprawdzenia.
     * @return true jeśli klucz istnieje.
     */
    boolean contains(K key);

}
```

7. Dodaj do pom.xml modułu zależność od biblioteki JUnit.
8. Zaimplementuj metody klasy MyMap implementującą inteface Map – na wzór java.util.Map.
9. Napisz testy jednostkowe wszystkich powyższych metod (przed dokonaniem pełnej implementacji tych metod!).
10. Uruchom testy poprzez Maven.
11. * Napisz testy dla metody tworzącej talię kart oraz tasującej talię tart z lab. 3.