## **Zadanie 1:** Clean Code

Klasa *pl.ttpsc.cleancode.doIt* jest daleka od ideału i w niewielkim stopniu przestrzega zasad Clean Code przedstawionych podczas wykładu.

Pytania uzupełniające:

a) Jaka jest rola tej klasy?
b) Jaką funkcjonalność realizuje?
c) Jaki jest najlepszy sposób na sprawdzenie poprawności implementacji kolejnych metod tej klasy? Upewnij się czy wszystkie metody są zaimplementowane poprawnie.

## **Zadanie 2:** Single Responsibility Principle

Reprezentacja modelu książki oraz możliwość jej drukowania (czytania) powinna być odseparowana. Dostarcz oddzielne implementacje modelu książki oraz jej drukowania zgodnie z *Single Responsibility Principle*.

Pakiet, w którym należy umieścić rozwiązanie: *pl.ttpsc.solid.srp.book.solution*

## Zadanie 3: Open-closed principle

Limity prędkości i kary za jej przekroczenie są różne dla każdego z 50 stanów USA. Należy dostarczyć łatwo rozszerzalną implementację, w której każdy stan będzie mógł zdefiniować swoje kary. Na chwilę obecną załóżmy, że będą to 3 stany USA, które skorzystają z naszego API, ale niewykluczone, że w przyszłości inne stany zdecydują się również skorzystać z API. Dlatego implementacja powinna być otwarta na rozszerzanie bez potrzeby zmieniania istniejącej już implementacji dla 3 stanów.

Pakiet, w którym należy umieścić rozwiązanie: *pl.ttpsc.solid.ocp.usa.solution*

## Zadanie 4: Liscov Substitution Principle

Podstawowym przykładem obrazującym zasadę Barbary Liskov jest zbudowanie aplikacji, która będzie obrazowała podstawowe figury geometryczne: kwadrat, prostokąd, koło... wraz z obliczaniem ich obwodu oraz pola powierzchni. Czy z punktu widzenia programisty i programowania obiektowego kwadrat jest prostokątem?

Pakiet w którym należy dostarczyć rozwiązanie: *pl.ttpsc.solid.lsp.shape*

## Zadanie 5: Interface Separation Principle

Klasa Contact ma dwie własności, jedna z nich przedstawia możliwość kontaktowania się przez wiadomość email, a druga to kontakt telefoniczny. Zaleca się wyróżnienie tych własności za pomocą interfejsów. Te interfejsy deklarują metody charakterystyczne dla danej własności.

Pakiet, w którym należy dostarczyć rozwiązanie: *pl.ttpsc.solid.isp.contactbook.solution*

## Zadanie 6: Dependency Inversion Principle

Mamy serwis informujący o zmianach pogodowych. Informacje o zmianach wysyłane są na różne urządzenia: telefony lub skrzynki poczty elektronicznej. Oba te urządzenia łączy jedna wspólna cecha jaką jest odbiór wiadomości i jej wyświetlenie. Telefon i skrzynka są modułami niskopoziomowymi, natomiast "Tracker" jest modułem wysokopoziomowym, który nie powinien zależeć od modułów niskopoziomowych.

Pakiet, w którym należy dostarczyć rozwiązanie: *pl.ttpsc.solid.dip.weathertracker.solution*