TDD

Test-Driven Development

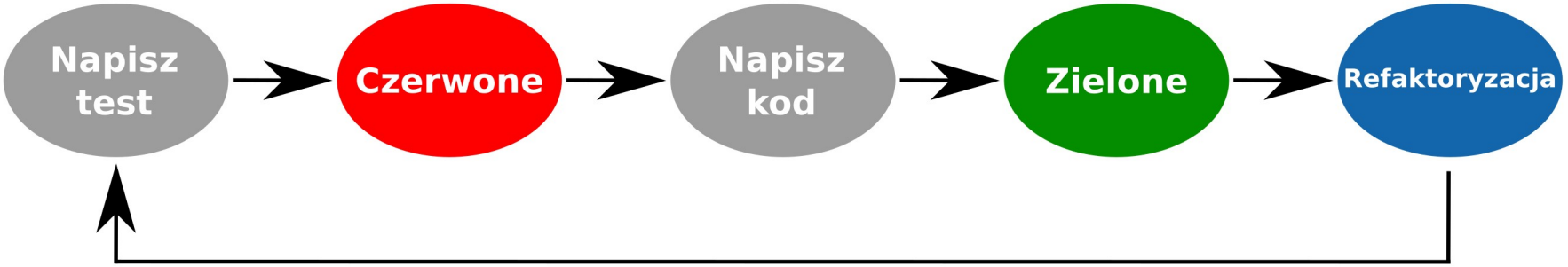
TDD

Idea

- Technika tworzenia oprogramowania polegająca na utworzeniu testu przed implementacją właściwej funkcjonalności – test powinien testować kod programu, który dopiero chcemy napisać
- Składa się z trzech etapów, tworzące jeden cykl:
  - Red
  - Green
  - Refactor
- Luźna analogia do gry w ping-ponga – następuje częste przełączanie się między kodem, a testami
- TDD opiera się o testy jednostkowe
- Zasady:
  - Nie można napisać kodu produkcyjnego, jeżeli nie napisano wcześniej nie przechodzącego testu
  - Nie można napisać większego testu niż jest to potrzebne, żeby ten test nie przeszedł. Kod, który się nie kompiluje, również oznacza nie przechodzący test
  - Nie można napisać więcej kodu produkcyjnego niż jest to potrzebne, żeby przeszedł test, który obecnie nie przechodzi

TDD

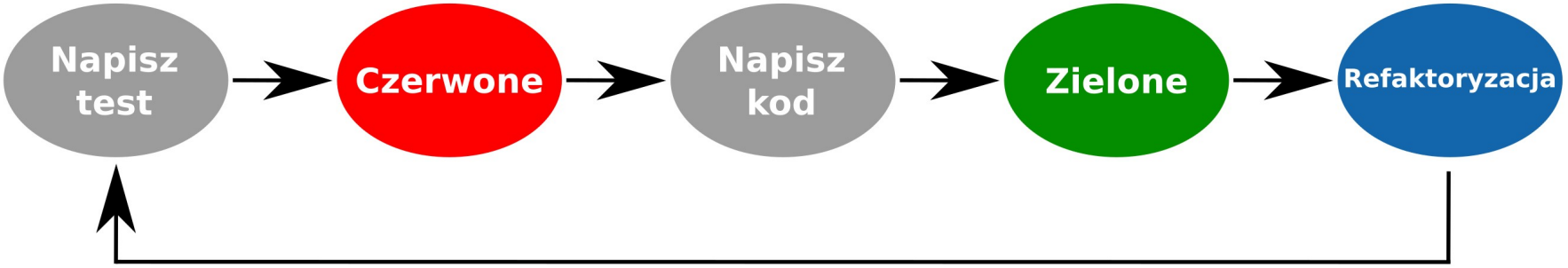
Cykl



- Napisz test
  - Projektujemy funkcjonalność aby dokładnie wiedzieć czego oczekujemy
  - Sprawdzamy w ten sposób zrozumienie problemu, który chcemy rozwiązać
  - Zmusza do zaplanowania struktury optymalnej, elastycznej i testowalnej

TDD

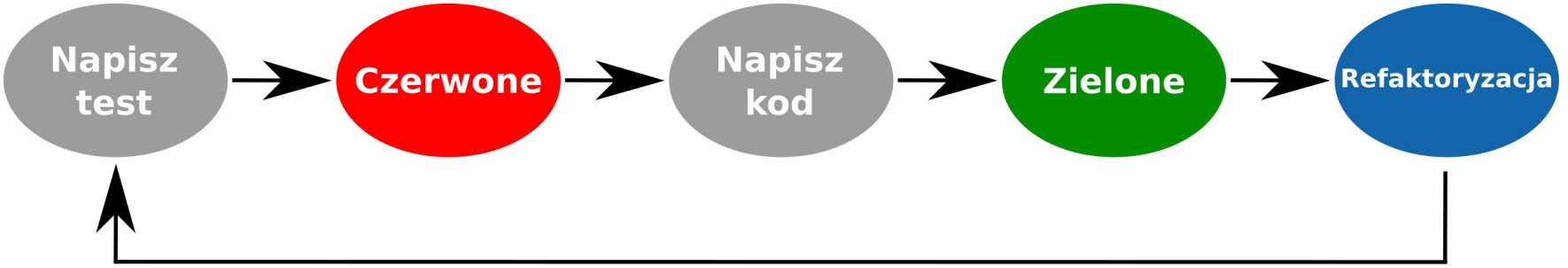
Cykl



- Faza Czerwone
  - Uruchomiony test kończy się niepowodzeniem (na czerwono) gdyż wynikają rozbieżności między oczekiwaniem z poziomu testów, a rzeczywistym kodem. Innymi słowy, nie istnieje kod zgodny z wymaganiami ostatniego testu.
  - Problematyczną oznaką jest poprawne zakończenie się nowo napasanego testu

TDD

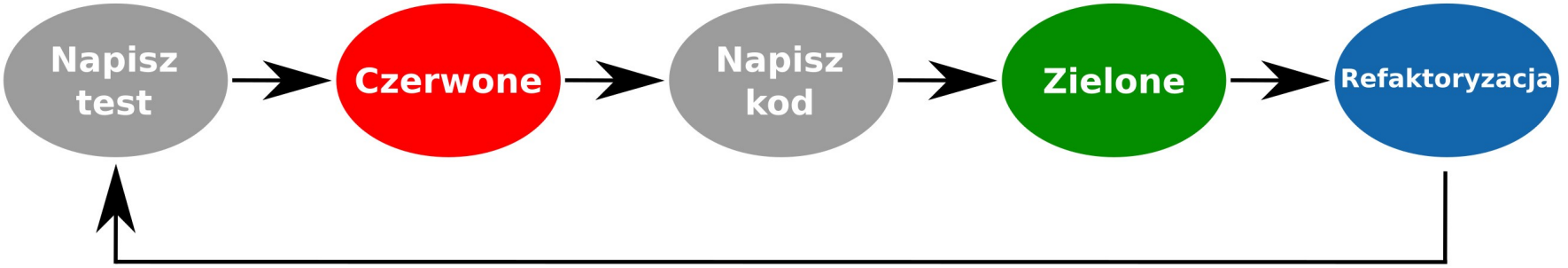
Cykl



- Napisz kod
  - Na tym etapie należy napisać kod umożliwiający przejście napisanego testu
  - Kod nie musi być optymalny i zgodny z dobrymi praktykami - jego celem jest zakończenie testu sukcesem.
  - Nie wskazane jest dodawać w tym miejscu nowych funkcji, których nie ma w testach. W przeciwnym razie należy przejść na sam początek cyklu

TDD

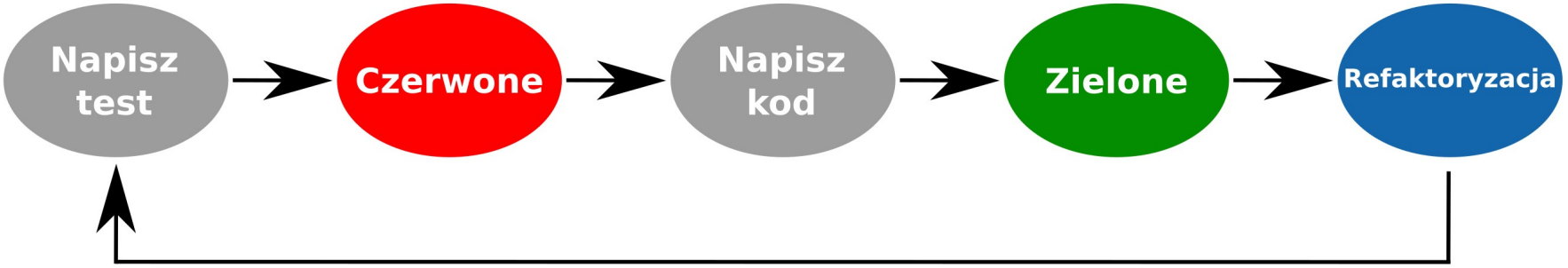
Cykl



- Faza Zielone
  - Uruchamiamy wszystkie testy – daje to nam pewność, że nowa funkcjonalność nie uszkodziła innych elementów programu
  - Celem tego etapu jest zakończenie wszystkich testów sukcesem (na zielono)
  - Dodatkowo badamy czas wykonywania wszystkich testów - jeżeli jest on niepokojąco długi może to oznaczać niską jakość testów lub zbyt dużo powiązań w samym kodzie

TDD

Cykl



- Faza Refaktoryzacja
  - Opcjonalny etap cyklu
  - Gdy uznamy, że funkcjonalność jest już gotowa, ale mało optymalna możemy przystąpić do tego etapu bez obaw, że naruszymy poprawność działania – posiadamy siatkę bezpieczeństwa w postaci testów

TDD

Gra FizzBuzz

- Pierwotnie jej celem była nauka dzieci dzielenia. Zabawa ta stała się również jedną z popularniejszych zadań programistycznych
- Zasady:
  - Wypisujemy kolejne liczby naturalne zaczynając od 1
  - Gdy mamy liczbę podzielną przez 3, wypisujemy Fizz, gdy mamy liczbę podzielną przez 5 wypisujemy buzz
  - Gdy liczba jest jednocześnie podzielna przez 3 i 5 wypisujemy FizzBuzz
  - W innych przypadkach wypisujemy podaną liczbę
- Przykład
  - Dla 1 → 1
  - Dla 2 → 2
  - Dla 3 → Fizz
  - Dla 4 → 4
  - Dla 5 → Buzz
  - Dla 6 → Fizz
  - Dla 15 → FizzBuzz

TDD
Gra Kółko - Krzyżyk

- Zasady:
  - Gra dla dwóch osób, oznaczanych symbolami X i O
  - Gracze na zmianę oznaczają pola w siatce 3x3
  - Wygrywa osoba, która umieści trzy swoje symbole obok siebie w pionie, poziomie lub po przekątnej
- Wymaganie 1:
  - Gdy symbol zostanie umieszczony poza osią X – zgłaszamy wyjątek
    - play_xOutsideBoard_returnException()
  - Gdy symbol zostanie umieszczony poza osią Y – zgłaszamy wyjątek
    - play_yOutsideBoard_returnException()
  - Gdy symbol zostanie umieszczony w miejscu innego – zgłaszamy wyjątek
    - play_occupiedBoard_returnException()

TDD
Gra Kółko - Krzyżyk

- Wymaganie 2:
  - Zaczynać powinien gracz używając symbolu X
    - nextPlayer_firstTurnWhenX_returnX()
  - Jeśli w poprzedniej kolejce dodano X, w następnej należy dodać O
    - nextPlayer_lastTurnWasXTehnO_returnO()
  - Jeśli w poprzedniej kolejce dodano O, w następnej należy dodać X
    - nextPlayer_lastTurnWasOTehnX_returnX()
- Wymaganie 3:
  - Wygrywa gracz, który pierwszy doda takie same symbole od krawedzi do krawędzi lub po przekątnej
    - play_noWinner_returnStringNoWinner()
    - play_winnerWhenFillHorizontaLine_returnStringWinner()
    - play_winnerWhenFillVerticalLine_returnStringWinner()
    - play_winnerWhenFillDianonallyLine_returnStringWinner()
- Wymaganie 4:
  - Wynik remisowy gdy wszystkie pola zostały zapełnione
    - play_allBoxesAreFilling_returnStringWinner_returnDraw()