**S (Single Responsible Principle)**

1. **Co to za zasada?**

S - **S**ingle Responsibility Principle, po Polsku zasada jednej odpowiedzialności, oznacza, że każda klasa powinna mieć tylko jedną odpowiedzialność. Jeżeli klasa ma więcej, niż jedną odpowiedzialność oznacza to powód do tego, aby ją w przyszłości zmienić.

1. **Jaki jest jej sens istnienia?**

Sensem zasady S jest ograniczenie ryzyka pojawienia się błędów poprzez skupienie się każdej klasy na jednym konkretnym zadaniu oraz zwiększenie czytelności, łatwości utrzymania i testowania kodu.

1. **Przykład kodu, który nie spełnia tej zasady:**

namespace ConsoleApp1

{

public class Rower

{

public string Model { get; set; }

public int Koło { get; set; }

public int Hamulce { get; set; }

public double kulkulatorizapisywacz()

{

double wartosc = 1000 + Koło + Hamulce;

return wartosc;

string zapisz = Model;

using (StreamWriter sw = new StreamWriter("C:\\tmp\\plik.txt"))

{

sw.WriteLine(zapisz);

}

}

}

internal class Program

{

static void Main(string[] args)

{

}

}

}

1. **Poprawiony kod zgodnie z zasadą S:**

namespace ConsoleApp1

{

public class Kalkulator

{

// tu liczymy wartość

}

public class Zapisywacz

{

// tu zapisujemy model

}

internal class Program

{

static void Main(string[] args)

{

}

}

}

1. **Przykład wzorca projektowego, który spełnia zasadę S**

Wzorzec adaptera spełnia zasadę S, ponieważ dostosowuje interfejs jednej klasy do interfejsu oczekiwanego przez inną, nie angażując się w dodatkowe funkcje.

1. **Przykład kodu z życia**

System biblioteki z wypożyczaniem i oddawaniem książek:

public class Wypozycz

{

public void Wypozycz(Ksiazka ksiazka, User user)

{

// wypożyczanie książki

}

public void Oddaj(Ksiazka ksiazka, User user)

{

// zwrot książki

}

}

public class Raport

{

public void Generuj(List<Ksiazka> books)

{

// Generowanie wypozyczen ksiazki

}

}

1. **Podsumowanie**

Zasada pojedynczej odpowiedzialności jest kluczowa dla tworzenia czytelnego, łatwego w utrzymaniu i elastycznego kodu. Stosowanie tej zasady prowadzi do poprawy jakości kodu i ułatwia współpracę nad projektem. Istnieją sytuacje, gdzie ścisłe trzymanie się tej zasady może być niewygodne lub niemożliwe. W takich przypadkach warto stosować zdrowy rozsądek i elastyczność.

**Źródła:**

-<https://www.programistabyc.pl/solid-1-srp/>

-<https://www.modestprogrammer.pl/solid-single-responsibility-principle-srp-wszystko-co-powinienes-wiedziec-o-zasadzie-pojedynczej-odpowiedzialnosci>

-<https://inetum.pl/zasady-solid-dobre-praktyki-w-programowaniu/#kod>