Notatki do 20211001

**Testowanie**

Najpopularniejsze biblioteki do testowania TestNG, JUnit 4, JUnit5 (tu przyklady z zajec)  
pom.xml - po utworzeniu projektu mavenowego to dodajemy zależnosci: (dependencies)  
  
metoda testowa moze byc public ale nie musi, nie zwraca wartosci - tylko void  
brak maina  
konwencja nazewnicza np klasa GreeterTest (test na koniec)  
nazwa metody np: shouldREturnCorrectSmth  
  
given - zalozenia poczatkowe, instancje obiektow  
when - wykonanie testu  
test - sprawdzzanie z oczekiwaniami (zwykle za pomoca asercji)  
assert equals / not equals , assertTRue/assertFalse, assertNull(result)/notNull  
asserSame(a1,a2) - do obiektów  
assertArrayEquals  
  
assertAll(  
() -> assertequals ...  
() -> assertequals ...  
() -> assertequals ...  
);  
sprawdzi np dlugosc tablicy, wartosc 1 elementu itd  
  
  
@test  
@DisplayName("jakis tekst, który sie wysietli zamiast nazwy metody")  
  
@Disabled - wyłączenie testu, nie zostanie uruchomiony  
  
  
metoda fail - failuje test (rzadka, ew zastosowanie pt aby nie zapomniec, że  
cos jeszcze jest do dopisania. np czekamy na kod od kogos, nie mozemy w danym  
momencie tego testu napisac

//given -inicjacja wszystkiego do testu  
nie zawsze musi byc  
  
assertThat - jak czerwony to  
alt enter i import  
  
assertThrows(ArithmeticException.class,  
 () -> calculator.divide(2, 0), "Cannot divide by 0");  
 z lambd ale poki co zapamietac  
  
 isInstanceOf(ArithmeticException.class); //do sprawdzenia typow  
  
task4  
  
ctrl lewy klik - podglad metod  
  
TDD - test driven development - najpierw testy potem dopisanie kodu (ale tylko dot, danego testu)  
np najpierw null i zeby był true, test passes to refactor  
//red (za 1 razem failed), green (implementacja aby wczesniejszy test byl pozytywny), blue  
  
//parametryzacja testów  
  
w metodach testowych raczej bez ifów i petli  
  
 @ParameterizedTest  
 void isBlank() {  
  
 input.equals("STOP") - tu mamy obiekt ktory moze byc nullem  
 i byc null pointer exception  
 ("STOP").equals.input - tu ok bedzie  
  
  
  
 ctrl shift kod dora/dół  
 nie gubi nawiasów

static int count = 0 -. moga byc modyfikowane przez inne metody. Wspoldzielona to dla  
innej moze byc juz np 5 bo pierwsza podbiła wczesniejszym wywolaniem  
int i=0 - osobna dla kazdego, dla kazdego od zera  
  
localDAte.of(1900,01,01)  
localDAte.parse("1900-01-01", DateTimeFormatter.iso\_date  
  
  
// Formater tworzymy - nowy, dla siebibe  
DateTimeFormatter formatter=DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");  
//LocalDate classesDate=LocalDate.parse(classesString, DateTimeFormatter.ISO\_LOCAL\_DATE);  
LocalDateTime nextClassesDate=LocalDateTime.parse(classesString, formatter);

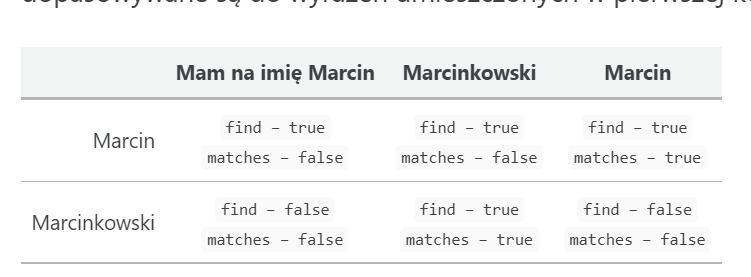
int[] array => int array[]  
  
teoretycznie dla dwuwym moznaby int[] array []  
  
if (csotam0 {  
continue; //zakonczenie iteracji, rozpoczecie nowej. Reszta kodu.  
continue przerywa iteracje, brak przerywa pętle  
}  
  
a[1]="X"  
a[2]="Y"  
  
  
for (String s: a) {  
sout (s + " "  
}  
  
null X Y  
  
String tekst2 = tekst.trim(); ///trim usuwa z przodu i z tyłu  
  
input.isBlank  
  
replaceall  
  
"happy"  
charAt -> okreslony znak 1 to "a"  
Substring -> od tego znaku daej 1 to "appy"  
  
  
name = ((city.getName() == null) ? "N/A" : city.getName())  
  
if (city.getName() != null) {  
 name = city.getName();  
} else {  
 name="N/A";  
}  
  
replace - zamienia kazdy  
replaceall - mozna uizyc regexa  
  
  
STRINGBUILDER  
  
/\* for (int x : tablica) {  
 System.out.println(x);  
 }\*/  
  
 static void Z17(int l1, int ... ln) {  
  
 }

sout("5+2 = " + 4 +3) --> 5+2 = 43  
sout(4+3+"5+2 = ") ---? 7+2 =

GIT

ls - zawartosc katalogu  
mkdir stworz katalog  
touch stworz pusty plik  
mv przenies  
mv (plik.txt) nazwakatalogu  
  
rm usun plik  
rm -r do usuniecia folderu  
cp skopiuj  
ls -la cala zawarto razem z ukrytymi  
ls --help  
bit branches  
status plikow ... utracked (nowy, zmiany nie sa sledzone), git add ,  
 status "changes to be commited" do zpaisu przy najblizszym commicie  
  
 git commit  
 git add .  
  
 vim plik.txt --- I .... potem :wq aby wyjsc (w write, q quit)  
  
git reset - wycofanie git add z pczekalni do working directory  
git checkout - przywraca do ostatniej postaci zapisanej przez git  
  
git rm usuniecie pliku  
wyciofanie usuniecia ze stage - git reset head nazwapliku.txt i potem jeszcze git checkout i nazwa pliku  
  
  
suma kontrolna SHA1  
  
git log - lista commitow  
  
-- patch zmiany w kazdym commicie  
-- stat  
-- oneline (w jednej linii)  
-- author="uzytkownik" --oneline --patch  
  
git log -3 (wyswietla 3 commity wstecz  
--graf historia w formie grafu  
--git log --oneline plik.txt (zmiany dot jednego pliku)  
wiecej w --help  
  
cofanie do wybranego commitu  
git checkout 611742c (ladujemy w odlaczonej glowie)  
powrot - git checkout master  
  
git revert <commit> - odwrocenie zmian w konternym comm  
git revert 1888da8 -> wycofanie tych zmian. zrobinowego commita  
  
zdalne repo  
  
git remote - nazwy  
-v dodatkowo url  
  
…or create a new repository on the command line  
echo "# Repo2021" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/AnnWitt/Repo2021.git  
git push -u origin main  
git6 remo  
…or push an existing repository from the command line  
git remote add origin https://github.com/AnnWitt/Repo2021.git  
git branch -M main  
git push -u origin main  
…or import code from another repository  
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.  
  
---------------------------Po synchro-------------  
plik .gitignoregi  
  
git branch  
git checkout (?)  
  
switch vs checkout  
  
  
Scalanie zmian, musimy być na branchu do ktorego vhcemy scalić zmiany (dobrac do niego zmiany)  
git merge NazwaBrancha  
  
usuwanie brancha  
git branch-d nazwa  
  
rebase/merge  
  
rebase zamienia // rebase-u mozliwosc zmia w commitach //mozliwosc okreslenia co bierzemy  
merge laczy  
  
  
stash - schowek, nie chcemy ich commitowac  
  
git stash   
list lista zapisanych zmian w schowky  
apply ponowne nalozenie zmiany, tylko itegtuje, nadal beda na liscie  
pop naklada zmiany i usuwa z listy  
  
//////////////////////////////////////////////////////////////////////////////////  
  
po ssh git@github.com:AnnWitt/Repo2021.git  
  
##Repozytoria zdalne + Laczenie z repozytorium zdalnym##  
wyświetlenie wszystkich skonfigurowanych serwerów -> git remote , git remote -v dodatkowo  
wyswietla źródło (ściezke)git  
  
https://gitlab.com/rafkapsa/java-podstawy/-/tree/main/out/production/Java%20podstawy/itneo  
  
…or create a new repository on the command line  
echo "# Projekt\_Zbiorczy" >> README.md  
git initikp  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/AnnWitt/Projekt\_Zbiorczy.git  
git push -u origin main //Origin to nazwa repo remote np git remote add test2Github  
https://github.com/AnnWitt/Repo2.git  
  
git remote add trenerPodstawy https://gitlab.com/rafkapsa/java-podstawy/-/tree/main/out/production/Java%20podstawy/itneo  
git remote add TrenerPodstawy https://gitlab.com/rafkapsa/java-podstawy/-/tree/main/  
  
…or push an existing repository from the command line  
git remote add origin https://github.com/AnnWitt/Projekt\_Zbiorczy.git  
git branch -M main (nie wiem czy potrzebne)  
git push -u origin main (nie wiem czy potrzebne)  
git status  
  
## Dodawanie usera ##  
$ git config --global user.name "Kursant SDA"  
$ git config --global user.email kursant@sda.pl  
git config --list ## podgląd  
  
## Tworzenie repozytorium ##  
Inicjalizacja Gita w istniejącym katalogu -> git init  
lonowanie istniejącego repozytorium -> git clone [url]  
  
##Stany plików w repozytorium##  
1. Sledzone - między innymi niezmodyfikowane, zmodyfikowane, w poczekalni  
2. Niesledzone - wszystkie pozostałe, nie znajdują się w poczekalni, są one gotowe do zatwierdzenia.  
  
##Poczekalnia zmiany##  
- rozpoczęcie śledzenia nowego pliku (przeniesienie pliku do poczekalni) -> git add nazwa\_pliku  
- commit - zatwierdzanie zmian z plików śledzonych -> git commit -m "treść wiadomości"  
- zatwierdzanie zmian z pominięciem poczekalni  
- usuwanie plików - wyrzucić z grupy plików śledzonych oraz następnie zatwierdzić zmiany (git rm)  
  
##Zmiana nazwy pliku#  
- git mv nazwa1.txt nazwa2.txt -> zmiana nazwy pliku  
  
##Lista zmian##  
- zwrócenie informacji, które dokładnie linie zostały dodane, a które usunięte: git diff  
- wyświetlenia szczegółowych informacji, które do poczekalni trafiły: it diff --staged  
  
##Wykluczenie plików z repozytorum -> .gitignore ##  
np: out/ (caly katalog),\*.txt , ddd.txt, rozpoczęcie "!" oznacza negację  
  
##Historia rewizji##  
git log  
git log --oneline --graph  
  
##Znak końca linii##  
https://devkr.pl/2019/07/25/lf-will-be-replaced-by-crlf/  
https://stackoverflow.com/questions/5834014/lf-will-be-replaced-by-crlf-in-git-what-is-that-and-is-it-important  
https://stackoverflow.com/questions/17628305/windows-git-warning-lf-will-be-replaced-by-crlf-is-that-warning-tail-backwar/38112001  
  
##Cofanie zmian (Zmiana nazwy)##  
-> Poprawienie commitu np nazwy albo pominietych plików: git commit --amend  
Mozna tez bez edytora git commit --amend -m "Nowy opis"  
Jeżeli jest już na serwerze: git push <remote> <branch> -f / --force (na siłę, force)  
- UWAGA -> Wysyłanie zmian metodą siłową powoduje nadpisanie gałęzi na serwerze waszą lokalną gałęzią.  
jeśli nie macie jakiegoś commit-a u siebie lokalnie bo kolega chwilę po was coś wysłał to zostanie to utracone.  
  
##Usuwanie plików z poczekalni##  
it reset HEAD <file> -> nie cofa zmian w zmodyfikowanym pliku, jedynie usuwa pliki z poczekalni  
  
##Cofanie zmian w zmodyfikowanym pliku##  
przywrócenie do stanu z ostaniego commita, w którym zostały zmodyfikowane. -> git checkout -- <file>  
  
##Tagi##  
Etykietowanie releasów. Wyswietlamy - git tag , Tworzymy git tag [nazwa-etykiety], git tag -a v1.4 -m 'my version 1.4' //z opisaniem etykiety  
np P1.1 , P1.2, P1.3 ,P2.1, P2.2 -> dwa tagi mozna zrobic P1 i P2  
wyszukanie git tag -l P1.\*  
Domyślnie, polecenie git push nie przesyła etykiet do zdalnego repozytorium. Trzeba je wysłać jak brancha -> git push origin [nazwa-etykiety].  
Etykietowanie starych commitów: git tag -a v1.2 9fceb02 (-a to etykieta opisowa, bez -m i'opis' otworzy się edytor)  
  
##Aliasy##  
Skroty polecen. UStawia się w pliku konfigruacyjnym np  
$ git config alias.co checkout  
$ git config alias.br branch  
potem się je wywoluje jako np co, br  
  
##Scalanie\pobieranie zmian z repozytoriów zdalnych"  
Pobieranie danych ze zdalnego repozytorium (ale bez scalenia) -> git fetch [remote-name]  
Jeżeli chcemy pobrać i scalić to -> git pull (wykonuje fetch a potem merge)  
wysyłanie zatwierdzonych zmian -> git push [nazwa-zdalnego-repo] [branch] np git push origin master  
  
##Zmiana nazwy/usuwanie repozytorium##  
Zmiana nazwy skrótu przypisanego do repozytorium zdalniego => git remote rename nazwa1 nazwa2  
Usunięcie odnośnika, skrótu, wszystkich referencji -> git remote rm nazwa2  
analogicznie usuwanie katalogow git rm katalog  
  
##Przegladanie repozytoriów##  
Dodatkowe info: -> git remote show [nazwa repo]  
gitgit  
##Branche/Gałęzie - podstawowe operacje##  
Tworzenie nowej -git> git branch nazwa (bez przelaczenia) lub git checkout -b nazwa\_gałęzi (od razu z przelaczeniem na ta galaz tj przesunieciem HEAD).  
HEAD info o galezi na ktorej znajduje sie uzytkownik. Sprawdzenie gdzie jest HEAD -> git log --oneline --decorate  
- Przelaczanie git checkout nazwa.  
- Lista lokalnych galezi: -> git branch , git branch -v doda info o aktualnych rewizjach, git branch -r wyswietli zdalne  
- scalenie: git branch --merged / git branch --no-merged -> lista galezi scalonych lub nie z aktywną galezia  
- usuwanie galezi -> git branch -d nazwa (trzeba byc na innej)  
  
##Scalanie\merge galezi##  
- scalenie - git merge -> wywolane z galezi docelowej (do niej chcemy domergowac). Np bedac na master robimy git merge nowafunkcjonalnosc  
- Scalenie Fast Forward: na galezi master utworzono np hotfix. Na hotfixie cos zrobiono i jest on domergowany do mastera. Jest "wspólny przodek", brak rozwidlen do scalenia  
- Scalenie trójstronne - gdy dochodzi do rozszczepienia galezi tj z mastera utworzono galaź na której coś zrobiono i zrobiono cos rownież na masterze. Master ma  
dodatkowy commit, którego nie ma nowoutworzonej galezi. Master w tym momencie nie jest już bezpośrednim "przodkiem"  
  
##Konflikty przy mergach (ten sam plik zmieniany na dwóch branchach##  
Jeżeli mamy jeden plik modyfikowany przez obie, zmiana nie zostanie zatwierdzona (caly proces wstrzymano). W git status pliki, które spowodowały konflikt są jako unmerged.  
Po otwarciu pliku mamy widoczne roznice do recznego skorygowania. Wtedy mozna puscic dalej  
  
##Powrot do starego commita##  
- powrót do starego commita -> git checkout <sha1>  
- modyfikować trzy ostatnie commity wybieramy : git rebase -i head~3 i otwiera się edytor w ktorym mozemy cos wykosic (pick/edit/squash)  
Usunięcie commita z listy spowoduje wykasowanie go (oczywiście wraz z wprowadzonymi przezeń zmianami) z historii.  
Przeniesienie w górę/w dół to modyfikacja jego miejsca w historii. Zmiana “pick” na “s” to doklejenie commita do poprzedniego  
(można jednocześnie przenieść commit w czasie i dokleić go do innego!). A “e” sprawi, że w proces modyfikacji zatrzyma się na etapie aplikowania  
tej właśnie zmiany dając nam możliwość dowolnej jej modyfikacji poprzez przedstawiony wyżej commit –amend.  
po zmianach -> git rebase --continue  
  
##Rebase##  
- git rebase master -> nasza aktualna gałąź, oryginalnie odchodząca od jakiegoś punktu w przeszłości gałęzi master, będzie od teraz  
widoczna jako odgałęzienie wyprowadzone z jej jej aktualnego HEADa.  
Można anulować poprzez git rebase --abort  
- git reset --hard orig\_head Powrót do stanu sprzed zakończonego właśnie rebase (dziala tez na merge)  
  
##IntelliJ problemy z ls -la##  
https://stackoverflow.com/questions/49032646/ls-is-not-recognized-as-an-internal-or-external-command-operable-program-or-b?rq=1  
używac dir zamiast ls -la  
  
##usuwanie repozytorium / katalogu ##  
rm -rf .git  
  
##GIT dodatkowe info##  
do git ignore walto dodac .idea/  
out/  
\*.iml  
---> bedzie dotyczyło kolejnych commitów, aby łapalo trzeba by najpierw rm na tych plikach zrobic  
  
  
https://java.pl.sdacademy.pro/e-podrecznik/git/git/

https://pl.wikipedia.org/wiki/ASCII  
  
wyjątek na wypadek niewlasciwego formatu danych od usera  
  
System.out.printf("Hello %s!%n", "World");  
This produces the following output:  
  
Hello World!  
As shown above, the format string contains plain text and two  
formatting rules. The first rule is used to format the string argument.  
The second rule adds a newline character to the end of the string.  
  
GIT ssh  
1. ssh-add -l  
2. ssh-keygen -t rsa -b 4096 -C "annwitt@tlen.pl"  
2. ssh-keygen -t rsa -b 4096 -C "annwa.wittmann@atos.net"  
3. /c/Users/Anna\_/.ssh/  
  
  
ssh-add -l  
ssh-keygen -t rsa -b 4096 -C "annwitt@tlen.pl"  
/c/Users/Anna\_/.ssh/  
  
C:\Users\Anna\_>ssh-keygen -t rsa -b 4096 -C "annwitt@tlen.pl"  
Generating public/private rsa key pair.  
Enter file in which to save the key (C:\Users\Anna\_/.ssh/id\_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in C:\Users\Anna\_/.ssh/id\_rsa.  
Your public key has been saved in C:\Users\Anna\_/.ssh/id\_rsa.pub.  
The key fingerprint is:  
SHA256:AZ16DsAkYo3kECp1Fvt9OHosQCFjgUJzKjhLdLUM9kM annwitt@tlen.pl  
The key's randomart image is:  
+---[RSA 4096]----+  
|\*OXOBE.. . |  
|X=\*BBo..o |  
|B+ o\* .. |  
|+o . .+..o |  
|. . .+S . |  
| . o.o |  
| o o |  
| o |  
| |  
+----[SHA256]-----+



short = 2 bajty 16 bitów od -32768 do +32768  
int 4 bajty  
long 8 bajtów 64 bity (czesto w bazach danych) (-2^63 - +2^63) dodajemy do liczby L np  
long zmiennaLong=10L;  
Long vs long - typ prymitywny a z wielkiej opakowane w obiekt. przy OBiekcie Long  
musi być to L, przy prymitywnym zadziała. Do obiektu można przypisać nulla, do prymitywnej  
nie mozna dodac wartosci nieoznaczonej (nulla)  
  
float/double typy zmiennoprzeciwnkowe  
float myFloatNumber=1.343424f; //f musi byc F lub f  
double myDoubleNumber=1.333d; //d musi byc D lub d  
  
 boolean myFalse=false;  
 boolean myTrue=true;  
 boolean myBoolean=myFalse && myTrue ; // true jeżeli obie będą true  
  
 && - and  
 || - or  
  
typ znakowy - char  
\t - tab  
\n - nowa linia  
\r - powrot karetki -> powrót do początku linii i nadpis (calości)  
  
char signValue='y';  
char tab='\t';  
np dp wyszukania znaku w kodzie  
  
String - typ łańcuchowy typ niemutowalny (nie moze byc zmieniany)  
  
operator += dodajemy podana wartosc i od razu przypisujemy  
i=10  
i+=20 //i =30  
  
% reszta z dzielenia  
  
post i preinkretementakcja  
int x=5;  
sout x++ ->5 wypisane - zwiekszenie po wypisaniu  
sput ++x ->6 zwiekszenie przed wypisaniem  
  
== rowne  
!= rozne  
! ->negacja np!(bot1||bot2)  
  
Konwersja typów  
niejawna - bbez konwersji , bez (int)  
int i =25 -> float j=i; int na float //cho tu moze byc strata przy duzych wartosciach  
  
rzutowanie - konwersja z mozliwoscia utraty (jawne z (int))  
(typ) zmienna np  
double n=99.98  
int m=(int) n; ---> 99  
int m2= (int) mMath.round(n) --->100  
  
instrukcje sterujace  
1f (warunek) {inst1} else {instr2}  
if (warunek1) {instr1} else if (warunek2) {instr}  
... elseif moze byc duzo ... else {instr3}  
  
switch (zmienna) {  
 case wartosc1  
 kod jezeli prawda  
 break; //bez break pojdzie dalej i wywola np defaulta  
 case wartosc2  
 default  
}  
  
petle  
for, while, dowhile  
  
for(deklaracja licznika; warunek; modyfikacja){}  
for (int i=0;i<10;i++) {cos}  
  
for (int i=1; i<=50;i=i+5)  
  
for each po tablicach  
 String[] imiona = {"Ala", "Adam", "Amdrzej","Ola","Pawel","Ewa"};  
 for (String imie : imiona) {  
 System.out.println("Imie: " + imie);  
  
  
 While (warunek) {instrukcja}  
  
 int i=0  
 while (i<10) { sout (i);  
 ++i;)  
  
  
do { instrukcja } while (warunek)  
wykona sie przynajmniej raz a nastepnie sprawdzi warunek  
 String[]imiona={"imie1", "imie2","imi3"} ;  
 int i= imiona.length;  
 int index=0;  
 do {  
 System.out.println(imiona[index]);  
 i++;  
 }  
 while (i< imiona.length);  
  
 warunek nigdy nie jest prawdziwy ale zdazył juz jeden raz wyswietlic  
  
 break; - przerwanie petli aby nie mieliło dalej  
  
 private static void przykladBrak() {  
 for (int i=0; i<10;i++) {  
 if (i==3) {break;} --> konczy  
 System.out.println(i);  
 }  
 }  
  
 private static void przykladContinue() {  
 for (int i=0; i<10;i++) {  
 if (i==8) {continue;} //wyswietli bez 8  
 System.out.println(i);  
 }  
 }  
  
Klasa String  
- literał String myTekstPrimit="tekst";  
- konstruktor String myTextkonstr = new String("text2"); //z tego mozna null przypisac, do prymitywnego nie  
  
klasa stirng niemutowalna - powstaje nowy string. nie posiada metoddy ()np setter)  
  
 System.out.println(text1==text2); //true  
 System.out.println(text2==text3); //false  
 System.out.println(text1==text3); //ctrl d na koncu wiersza //false  
 System.out.println(text1.equals(text2)); //true  
 System.out.println(text1.equals(text3)); //true  
  
 == -? equals -> roznica - w 1 czy ta sama referencja, druga sprawdzanie stricte wartosci  
 zasadniczo dla Stringów korzystamy z equals dla porównan  
  
API serwis zewnętrzny - googlnij restful api  
  
 //String - konkatenacjia napisów  
  
 System.out.println(text1+text2);  
 .... (text.concat(text2))  
  
 indexof - pozycja pierwszego wystapienia iagu no test.indexof("is")  
  
 text.replaceAll("a","o")) podmiana a na o  
  
  
Wejscie/wyjscie  
  
Scanner skan=new Scanner(System.in)  
String textwczutany=skan.nextline()  
nextInt, Double, Boolean  
  
printf() z formatowaniem  
  
e,f,x,o,s,c,b np printf(44,2) z 2 miejscami po przecinku  
  
Metody - grupy instrukcji  
moze zawierac argumenty lub nie  
int diff(int arg1,int arg2) {  
return arg1-arg2;  
}  
mozna zamiast listy wybrac obiekt  
  
void nie zwraca nic, jak no int to musi byc return  
  
void + return to np return jezeli jakis warunek spelniony. Return pelni funkcję breaka  
  
Tablice  
  
jedno/wielowymiarowe - rozmiar powinien byc staly  
  
typ[] nazwa=new typ[liczba elementow]  
  
String[] array=new String[]{"w1","w2,"w3"}  
  
array[2]="ddd"; (dp 3 komorki tablicy)  
  
array.lenght - wierszy  
arrau[i].lenght - ile kolumn  
  
tablice z tablicami (obiekty)  
  
typ[][]tablicaWiel  
tablicaWiel= new typ[l wierszy][l kolumnn] np new String[2][];  
tablicaWiel[0]=new String[]{"ddd","www"}; pierwszy wiersz  
tablicaWiel[1]=new String[]{"dd2d","w2ww"}; drugi wiersz  
  
Inny zapis warunku  
return foundNumbers.isEmpty() ? Collections.singletonList(-1) : foundnumbers  
  
if (foundNumbers.isEmpty()) {  
 return Collections.singletonList(-1) ;  
} else {  
 return foundnumbers;  
}  
  
  
 if (liczba1>liczba2) {  
 m1=liczba1;  
 } else {  
 m1=liczba2;  
 }  
  
 m2 = liczba3>liczba4 ? liczba3: liczba4;  
  
UWAGI - ENUM/CLASS ROZNICA  
  
  
TABLICA WYWOLANA Z METODY DO UZYCIA  
 static int[] tablica () {  
 int[] tblTest=new int[]{3,5,6,7,2};  
 return tblTest;  
 };  
  
 A MAIN  
 int[] x=tablica();  
 System.out.println(x[2]);  
  
  
  
 Klsy/obiekty  
  
 pole - zmienna, cecha klasy  
 metoda operacja  
 klasa schemat na ktory zdefiniowane sa pola i metody  
  
  
  
package z20210821;  
  
public class samochod {  
 private String color="Czerwony";  
 public String getColor(){  
 return color;  
 }  
 public void setColor (String color) { //kolor wprowadzony  
 this.color = color; //this to czerwony (mimo, że ta sama nazwa zmiennej)  
 }  
  
}  
  
 public void setColor (String colorWprowadzony) { //kolor wprowadzony  
  
 this.color = colorWprowadzony; //this to czerwony  
 }  
  
 modyfikatory dostepu [modyf] xx class {}  
 public - dostep z kazdego miejsca aplikacji  
 protected -  
 default - package private (gdy nic nie wpiszemy) - dziala w ramach pakietu, poza nim nie  
 private  
  
 obiwekty=instancje klasy  
gettery i settery tez moga miec modyfikatory  
  
 new wywoluje tzw kontruktor (Car car1=new Car()  
  
enkapsulacja - zamykanie pol w prywatnych i dobieranie sie do nich getterami i setterami (pola w miare mozliwosci powinny byc prywatne  
  
protected - w ramach pakietu,  
  
PAKIETY  
 package pl.sdaa.example //odwrocona nazwa domeny // mozna te same nazwy klas w roznych pakietach  
 rozdzielamy kropkami, powinna odpowiadac strikturze na dysku  
  
  
 importowanie klas  
  
 impost nazwa.pakietu.klasa;  
 ---------------------  
 import dwoch las po tej samej nazwie  
  
 import pl.sda.xxx.secondex  
  
 Book testBook=new Book()  
  
 pl.sda.xxx.secondex.Book secondTestBook= new ....  
  
 --------  
  
 import staqtyczny  
  
 public static void //statyczne  
  
 import static java.lang.Math.PI;  
 wtedy można to wywołać samym PI  
 Pomocne przy stałych bo nie trzeba MAth.PI  
  
  
 KONSTRUKTORY  
  
 SPECJALNY RODZAJ METODY DO INICJALIZOWANIA STANU OBIEKTU (WARTOSCI POCZATKOWE)  
 NAZWA KONTRUKTORA IDENTYCZNA Z NAZWĄ KLASY  
 NIC NIE ZWraca (nawet nie void), brak deklaracji zwrcaneog typi (np int)  
 przyjmuje dowolną liczbę elemenów wejsciowych  
  
konstruktor zwykle publiczny ale może być prywatny  
jezeli brak konsturktora to jest bezparametrowy, domyuslny  
  
fklasa moze byc tez final zamiennie to private  
  
przeciazenie konstruktorow - aby miec wiecej o tej samej nazwie ale musi miec rozna ilosc parametrow  
  
przy statycznej nie jest koniecznie utworzenie obiektu  
wystarczy odwolac sie bezposrednio do klasy  
  
  
klasy wewnętrzne - klasa w klasie Myouterklas.myinnerklas .....  
  
Myouterklas nazwa=new myouterclass()  
  
int vs Integer - ten drugi moze miec nulla  
  
throw new RuntimeException()  
  
public class lock {  
private int kod  
public void przekladnia () {  
kod++; (zwiekszanie o 1 ... dlatego nie moze byc final  
}  
  
}  
  
--------------  
  
Varargs - zmienne liczb y argumentow tego samego typu  
  
----- wyrazenia regularne  
  
regex - czy dane maja poprawny format  
  
z sekwencji atomow  
literał - literac cyfra itd  
kwantyfikatory - liczba wystapien atomu  
  
zakres znakow albo w kwadraotywch bez przecinkow [abc]  
abo przedział [1-3;0-9]  
  
implementujemy klasami Pattern i Matcher  
  
Pattern pattern=Pattern.compile("a+bcd")  
Matcher matcher=pattern.macher  
 Na przykład wyrażenie regularne \d{2}-\d{3} opisuje zbiór kodów pocztowych w Polsce.  
  
Klasa Matcher posiada, między innymi, następujące metody:  
  
find() – metoda zwraca true jeśli w łańcuchu znaków znajduje się coś co pasuje do wyrażenia regularnego,  
matches() – metoda zwraca true jeśli łańcuch znaków pasuje w całości do wyrażenia regularnego.  
https://www.samouczekprogramisty.pl/wyrazenia-regularne-w-jezyku-java/  
  
wyrażenia regularne mogą zawierać zwykłe literały znakowe, na przykład kot czy pies to poprawne wyrażenie regularne.  
Jednak są znaki, które interpretowane są w specjalny sposób. Wyrażenie takie jak 1 + 2 = 3 zawiera jeden ze znaków  
specjalnych – znak +. W związku z tym, jeśli znak + chcemy interpretować dosłownie musimy poprzedzić go znakiem \, wówczas  
pomijamy jego specjalne znaczenie. W takim przypadku otrzymasz wyrażenie regularne 1 \+ 2 = 3.  
  
W języku Java wyrażenia regularne zapisujemy używając typu String. Znak \ jest w literałach znakowych traktowany  
specjalnie (podobnie jak w samych wyrażeniach regularnych). Na przykład literał znakowy "\t" oznacza znak tabulacji,  
więc przy zapisie "1 \+ 2 = 3" kompilator Javy doszukiwałby się specjalnego znaczenia dla "\+" (podobnie jak przy "\t") a  
nie o to nam tutaj chodzi. Dlatego właśnie w języku Java w wyrażeniach regularnych musimy „dublować” każdy ukośnik  
  
Znak ? oznacza – element znajdujący się wcześniej jest opcjonalny. Innymi słowy to co występuje przed ? może wystąpić  
raz lub może zostać pominięte. Na przykład do wyrażenia regularnego kr?at pasują zarówno "krat" jak i "kat" ale nie  
pasuje "kot" czy "krrat".  
  
Znak \* oznacza – powtórz dowolną liczbę razy element znajdujący się wcześnie  
  
Znak + jest podobny do \*. Oznacza on, że występujący przed nim element musi być powtórzony 1 lub więcej razy  
  
Poza znakami ?, + i \*, które określają dopuszczalną liczbę powtórzeń możesz też użyć {}. Jednak {} ma większe możliwości:  
{x} – oznacza że element poprzedzający musi wystąpić dokładnie x razy,  
{x,} – oznacza, że element poprzedzający musi wystąpić co najmniej x razy,  
{x,y} – oznacza, że element poprzedzający musi wystąpić od x do y razy.  
Zauważ, że symbole ?, \* i + możemy zastąpić {}. Na przykład wyrażenia regularne al\*a i al{0,}a czy al+a i al{1,}a są  
sobie równoznaczne. Jednak zapis z ?, + czy \* jest krótszy przez co częściej stosowany.  
  
np a{2,3}b{4,9}c{1,9} itd  
  
  
Znak .Permalink  
Znak oznacza dowolny symbol (poza znakiem nowej linii). Innymi słowy do wyrażenia regularnego ko.ek pasują zarówno  
 "kotek" jak i "korek" ale nie pasuje "koek" czy "ktek".  
  
  
 Takie często używane klasy zostały wbudowane w wyrażenia regularne pod postacią predefiniowanych klas. Wszystkie predefiniowane klasy prezentuje lista poniżej:  
 \d – jakakolwiek cyfra [0-9],  
 \D – jakikolwiek znak, który nie jest cyfrą [^0-9],  
 \w – znak używany w słowach [a-zA-Z0-9\_] (zauważ, że mamy tu znak \_),  
 \W – jakikolwiek znak, który nie jest używany w słowach [^a-zA-Z0-9\_],  
 \s – tak zwane białe znaki czyli znak spacji czy tabulacji [ \t\n\r\f\x0B]. Możesz je opisać jako znaki, które nie są widoczne podczas wydruku,  
 \S – negacja grupy \s czyli [^ \t\n\r\f\x0B]  
  
  
 String ciagZnakowWzorzec="[a-z]+";  
 Pattern wzorzec=Pattern.compile(ciagZnakowWzorzec);  
 Matcher dopasowanie= wzorzec.matcher(ciagZnakow);  
 System.out.println(" Dla ciągu znaków " + ciagZnakow + ": "+ dopasowanie.matches());