

Nazwa kwalifikacji: **Projektowanie, programowanie i testowanie aplikacji**

Symbol kwalifikacji: **INF.04**

Numer zadania: **01**

Wersja arkusza: **SG**

Wypełnia zdający

Numer PESEL zdającego*

--	--	--	--	--	--	--	--	--	--	--	--

Numer stanowiska

--	--	--

Miejsce na naklejkę z numerem
PESEL i z kodem ośrodka

Czas trwania egzaminu: **180** minut

INF.04-01-26.01-SG

EGZAMIN ZAWODOWY

Rok 2026

CZĘŚĆ PRAKTYCZNA

**PODSTAWA PROGRAMOWA
2019**

Instrukcja dla zdającego

1. Na pierwszej stronie arkusza egzaminacyjnego wpisz w oznaczonym miejscu swój numer PESEL*, numer stanowiska i naklej naklejkę** z numerem PESEL i z kodem ośrodka.
2. Sprawdź, czy arkusz egzaminacyjny zawiera 6 stron i nie zawiera błędów. Ewentualny brak stron lub inne usterki zgłoś przez podniesienie ręki przewodniczącemu zespołu nadzorującego.
3. Zapoznaj się z treścią zadania oraz stanowiskiem egzaminacyjnym. Masz na to 10 minut. Czas ten nie jest wliczany do czasu trwania egzaminu.
4. Czas rozpoczęcia i zakończenia pracy zapisze w widocznym miejscu przewodniczący zespołu nadzorującego.
5. Wykonaj samodzielnie zadanie egzaminacyjne. Przestrzegaj zasad bezpieczeństwa i organizacji pracy.
6. Po zakończeniu wykonania zadania pozostaw arkusz egzaminacyjny z rezultatami wykonania zadania na swoim stanowisku lub w miejscu wskazanym przez przewodniczącego zespołu nadzorującego.
7. Po uzyskaniu zgody zespołu nadzorującego możesz opuścić salę/miejsce przeprowadzania egzaminu.

Powodzenia!

* w przypadku braku numeru PESEL – seria i numer paszportu lub innego dokumentu potwierdzającego tożsamość

** w przypadku otrzymania naklejki

Zadanie egzaminacyjne

UWAGA: numer, którym został podpisany arkusz egzaminacyjny (PESEL lub w przypadku jego braku, numer paszportu) jest w zadaniu nazywany numerem zdającego.

Wykonaj aplikację konsolową oraz mobilną według wskazań. Wykonaj testy jednostkowe aplikacji konsolowej. Wykorzystaj konto **Egzamin** bez hasła. Do wykonania aplikacji mobilnej wykorzystaj umieszczone na pulpicie konta **Egzamin** obrazy z archiwum *zad1.7z* z hasłem: **@Kosc6**

Utwórz folder i nazwij go numerem zdającego. W folderze utwórz podfoldery: *konsolowa*, *mobilna*, *testy*.

Część I. Aplikacja konsolowa

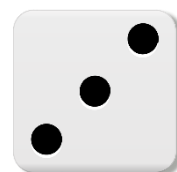
Zaprogramuj klasę o nazwie *Kosc* implementującą logikę działania pojedynczej kości, wykorzystywanej później w grze w kości. Każda ściana kości zawiera unikatową liczbę oczek równą wartości wyrzuconej z zakresu $1 \div 6$.

Założenia do programu:

- Wykonywany w konsoli
- Zastosowany obiektowy język programowania zgodny z zainstalowanym na stanowisku egzaminacyjnym: C++ lub C#, lub Java, lub Python
- Zastosowane znaczące, angielskie lub polskie nazewnictwo zmiennych i funkcji
- Zapisany czytelnie, z zachowaniem zasad czystego formatowania kodu

Klasa *Kosc* powinna zawierać:

- Pole ogólnodostępne, statyczne, przechowujące liczbę instancji klasy *Kosc*
- Pola ogólnodostępne, przechowujące:
 - Nazwy plików przechowujących obrazy (*kosc0.png*, *kosc1.png*, *kosc2.png*, *kosc3.png*, *kosc4.png*, *kosc5.png*, *kosc6.png*) umieszczone w tablicy lub innej kolekcji typu napisowego
 - Liczba oczek wyrzucona kością, typu liczbowego całkowitego (3 dla obrazu 1)
 - Identyfikator pliku graficznego odpowiadającego wyrzuconej liczbie oczek, typu całkowitego (3 dla obrazu 1, jest to indeks tablicy wskazujący na nazwę *kosc3.png*)
 - Informacja czy kość jest dostępna, typu logicznego
- Konstruktory klasy:
 - Jednoargumentowy, którego argument jest wartością wyrzuconej kości.
 - W przypadku, gdy wartość argumentu jest inna niż 1, 2, 3, 4, 5 lub 6, ustawia wartość na 0.
 - Przypisuje liczbie oczek i identyfikatorowi pliku wartość argumentu
 - Przypisuje wartość polu logicznemu: kość jest dostępna
 - Inkrementuje zmienną statyczną zliczającą instancje klasy
 - Bezargumentowy:
 - Losuje liczbę pseudolosową z zakresu od 1 do 6
 - Przypisuje liczbie oczek i identyfikatorowi pliku wylosowaną liczbę
 - Przypisuje wartość polu logicznemu: kość jest dostępna
 - Inkrementuje zmienną statyczną zliczającą instancje klasy



Obraz 1.
kosc3.png

Uwaga: W języku Python należy utworzyć jeden konstruktor z domyślną wartością argumentu *None*

- Metoda ogólnodostępna, bezparametrowa, niezwracająca wartości, która realizuje rzut kością tylko, gdy kość jest dostępna:
 - Losuje wartość z zakresu od 1 do 6
 - Przypisuje wylosowaną wartość do pola liczby oczek i identyfikatora pliku graficznego
- Metoda ogólnodostępna, bezparametrowa, niezwracająca wartości, która blokuje kość:
 - Ustawiana jest informacja, że kość jest niedostępna
- Metoda ogólnodostępna, bezparametrowa, zwracająca wartość wyrzuconą na kości w postaci tekstu (np. gdy wartość jest równa 3, zwracane jest „trzy”)

Sprawdź działanie klasy w programie głównym:

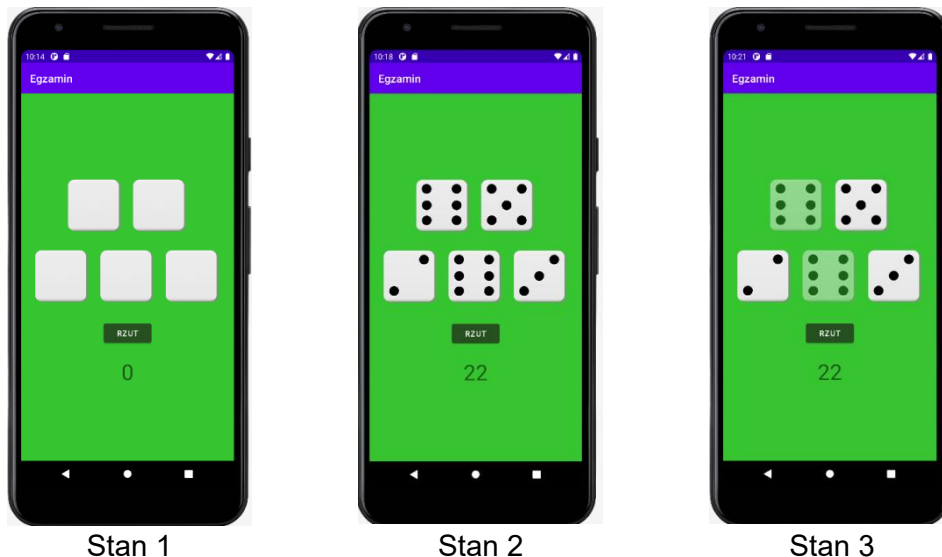
- Należy utworzyć dwa obiekty klasy *Kosc*, każdy za pomocą innego konstruktora
- W przypadku konstruktora jednoargumentowego liczba przekazana jako argument ma być pobrana z klawiatury
- Po utworzeniu każdego obiektu należy wyświetlić:
 - Liczbę utworzonych instancji klasy
 - Informację o liczbie oczek wyrzuconych kością (w postaci liczbowej i napisowej)
 - Nazwę pliku odpowiadającego wyrzuconej liczbie oczek
- Informacja powinna być zrozumiała dla użytkownika

Wykonaj zrzuty ekranu dokumentujące wykonanie programu w konsoli. Zrzuty ekranu powinny obejmować cały obszar ekranu, z widocznym paskiem zadań oraz środowiskiem programistycznym. Liczba zrzutów ekranu powinna odpowiadać wszystkim możliwym interakcjom użytkownika z programem. Zrzuty ekranu zapisz w folderze *konsolowa* pod nazwami *konsola1.png*, *konsola2.png*, itd.

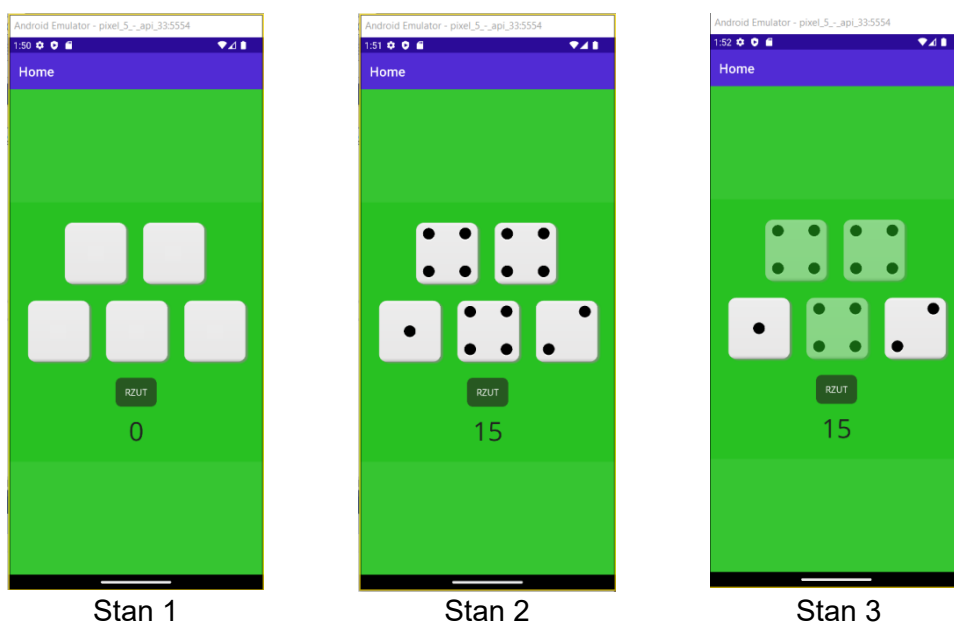
Kod aplikacji przygotuj do nagrania na płytę. W folderze *konsolowa* powinno znaleźć się archiwum całego projektu o nazwie *konsola.zip*, skopiowany z projektu plik z kodem źródłowym programu, plik wykonywalny, jeżeli istnieje oraz zrzuty ekranu.

Część II. Aplikacja mobilna

Za pomocą środowiska programistycznego dostępnego na stanowisku egzaminacyjnym wykonaj aplikację mobilną do gry w kości oraz uruchom ją w dostępnym emulatorze systemu mobilnego. Gra polega na rzutach pięcioma sześciennymi kośćmi. Stany aplikacji zostały przedstawione na obrazach 2 i 3. W zależności od zastosowanego środowiska programistycznego oraz emulowanego systemu, widok może nieznacznie się różnić. W programie można wykorzystać klasę *Kosc* z aplikacji konsolowej.



Obraz 2. Stany aplikacji mobilnej w Android Studio



Obraz 3. Stany aplikacji mobilnej w .NET MAUI lub XAMARIN

Elementy aplikacji w stanie początkowym (stan 1 obrazu 2 lub 3):

- Pięć obrazów *kosc0.png* bez przezroczystości (wszystkie kości są dostępne)
- Przycisk z napisem „RZUT”
- Pole tekstowe z wynikiem rzutu z wartością 0, zawierające sumę oczek pięciu kości
- Rozmieszczenie elementów zgodne z obrazami 2 lub 3

Założenia dotyczące widoku:

- Interfejs użytkownika zapisany za pomocą języka znaczników wspieranego w danym środowisku (np. XAML, XML)
- Zastosowany dowolny układ pozwalający na rozmieszczenie wszystkich elementów tak jak na obrazie 2 lub 3
- Tło okna / rozkładu: #ED27C121, tło przycisku: #ED275021
- Rozmiar czcionki pola tekstowego 40
- Wielkość obrazów wyświetlających kości dopasowana do widoku z obrazu 2 lub 3
- Marginesy zewnętrzne dla obrazów i przycisku 10
- Wszystkie elementy widoku są wyśrodkowane

Działanie aplikacji po wciśnięciu przycisku RZUT:

- Dla dostępnych kości losowana jest liczba oczek każdej z nich
- W kontrolkach obrazu wyświetlana jest grafika odpowiadająca liczbie oczek na każdej kości
- Liczona jest suma oczek z wszystkich kości i wyświetlana w polu tekstowym

Działanie aplikacji po kliknięciu na dowolny obraz kości:

- Jeżeli kość jest dostępna: jest ustawiana na niedostępną i przezroczystość obrazu jest ustawiana na 50%
- Jeżeli kość jest niedostępna: jest ustawiana na dostępną i obraz jest wyświetlany bez przezroczystości

Przykłady interakcji:

- Obraz 3, stan 2: ponieważ wszystkie kości są dostępne, jest losowanych 5 wartości (4, 4, 1, 4, 2), wyświetlane są odpowiadające im grafiki (*kosc4.png*, *kosc4.png*, *kosc1.png*, *kosc4.png*, *kosc2.png*) oraz liczona jest suma oczek równa 15 ($4 + 4 + 1 + 4 + 2 = 15$)
- Obraz 3, stan 3: zostały kliknięte kości: pierwsza, druga i czwarta (4, 4, 4); została zmieniona ich przezroczystość, są niedostępne

Aplikacja powinna być zapisana czytelnie, z zasadami czystego formatowania kodu, należy stosować znaczące nazwy zmiennych i funkcji.

Podejmij próbę kompilacji i emulacji aplikacji. Wykonaj zrzuty ekranowe dokumentujące wszystkie stany aplikacji. Zrzuty zapisz w folderze *mobilna* pod nazwami *mobilna1.png*, *mobilna2.png*, itd. Wszystkie zrzuty muszą zawierać cały obszar ekranu z paskiem zadań i środowiskiem programistycznym.

Kod aplikacji przygotuj do nagrania na płytę. W folderze *mobilna* powinno znaleźć się archiwum całego folderu projektu oraz skopiowane pliki z kodem źródłowym, które były modyfikowane oraz pliki z zrzutami.

Część III. Testy jednostkowe

Wykonaj testy jednostkowe aplikacji konsolowej, metody realizującej rzut kością. Do napisania testów wykorzystaj bibliotekę do testów jednostkowych dostępną na stanowisku egzaminacyjnym dla języka programowania, w którym została napisana klasa *Kosc*. W testach jednostkowych należy rozważyć następujące przypadki testowe:

- Czy wyrzucona wartość jest w zakresie od 1 do 6
- Czy wartość na kości pozostaje bez zmian jeżeli kość nie jest dostępna

W testach nazwy metod testujących należy dobrać tak, aby wyrażały cel danego testu. Nazwy metod należy zapisać zgodnie z konwencją nazewnictwa w danym języku programowania. Każdy przypadek testowy powinien być sprawdzany oddzielną metodą.

Po napisaniu metod testujących uruchom wszystkie metody. Wykonaj zrzuty ekranu dokumentujące uruchomienie testów, zrzuty zapisz w katalogu *testy* pod nazwami *test1.png*, *test2.png*, itd. Na zrzutach powinny być widoczne wyniki działania metod testujących. Zrzuty ekranu powinny obejmować cały obszar ekranu, z widocznym paskiem zadań oraz środowiskiem programistycznym.

Projekt testów przygotuj do nagrania na płytę. W folderze *testy* powinno znaleźć się archiwum całego folderu projektu oraz skopiowane pliki z kodem źródłowym, które były modyfikowane oraz zrzuty ekranu.

Utwórz plik tekstowy z dokumentacją i nazwij go *egzamin*. Dokument powinien zawierać informacje o narzędziach wykorzystanych na egzaminie:

- Nazwę systemu operacyjnego
- Nazwy środowisk programistycznych
- Nazwę emulatora dla aplikacji mobilnej
- Nazwy języków programowania

Dokument *egzamin* zapisz w folderze z numerem zdającego.

UWAGA: Nagraj płytę z rezultatami pracy. W folderze z numerem zdającego powinny się znajdować foldery: konsolowa, mobilna, testy oraz plik egzamin. W folderze konsolowa: spakowany cały projekt aplikacji konsolowej, pliki źródłowe, plik wykonywalny jeśli istnieje, zrzuty ekranu. W folderze mobilna: spakowany cały projekt aplikacji mobilnej, pliki z kodem źródłowym interfejsu i logiki, zrzuty ekranu. W folderze testy: spakowany projekt z testami jednostkowymi, pliki z zapisem metod testujących, zrzuty ekranu. Po nagraniu płyty sprawdź poprawność nagrania. Opisz płytę numerem zdającego i pozostaw na stanowisku, zapakowaną w pudełku wraz z arkuszem egzaminacyjnym.

Czas przeznaczony na wykonanie zadania wynosi 180 minut.

Ocenie podlegać będą 4 rezultaty:

- implementacja, kompilacja, uruchomienie programów,
- aplikacja konsolowa,
- aplikacja mobilna,
- testy aplikacji.

Wypełnia zdający

Do arkusza egzaminacyjnego dołączam płytę CD/DVD opisaną numerem PESEL

--	--	--	--	--	--	--	--	--	--	--

, której jakość nagrania została przeze mnie sprawdzona.

PESEL

--	--	--	--	--	--	--	--	--	--

, której jakość nagrania została przeze mnie sprawdzona.

[illegible]

Wypełnia Przewodniczący ZN

Potwierdzam, że do arkusza egzaminacyjnego dołączona jest płyta CD/DVD, opisana numerem PESEL zdającego.

.....
Czytelny podpis Przewodniczącego ZN

.....
Czytelny podpis Przewodniczącego ZN