Nazwa kwalifikacji:	**Projektowanie, programowanie i testowanie aplikacji**
Oznaczenie kwalifikacji:	**INF.04**
Numer zadania:	**01**
Kod arkusza:	**INF.04-01-26.01-SG**
Wersja arkusza:	**SG**

Lp.	Elementy podlegające ocenie/kryteria oceny
R.1	**Rezultat 1: Implementacja, kompilacja, uruchomienie programu**
	Uwaga: kryteria należy odnieść do aplikacji konsolowej, jeżeli ta nie istnieje, kryteria zastosować do aplikacji mobilnej. Kryteria dotyczą wyłącznie samodzielnie napisanego kodu. *Wystarczy, że sprawdzaną cechę zastosowano dla większości (90%) przypadków w kodzie*
R.1.1	Kod źródłowy zapisany w sposób czytelny: instrukcje w osobnych liniach, stosowane spacje pomiędzy operatorami, konsekwentnie stosowana wybrana konwencja dla nawiasów klamrowych instrukcji blokowej
R.1.2	Kod zapisany z wcięciami dla zagnieżdżeń bloków
R.1.3	Polskie lub angielskie nazewnictwo metod. Nazewnictwo jest znaczące
R.1.4	Polskie lub angielskie nazewnictwo pól i zmiennych. Nazewnictwo jest znaczące. Wyjątkami od reguły są zmienne bufor, tmp, iteratory pętli. Kryterium nie jest spełnione, gdy nazwy zmiennych nic nie znaczą, np. x, tab, tablica, foo
R.1.5	Typy zmiennych pasują do problemu, np. tablica przechowuje napisy, liczba oczek oraz identyfiaktor pliku graficznego są typu całkowitego, zmienna przechowująca czy kość jest dostępna jest typu logicznego *(w przypadku języka Python typ wynika z przypisanych danych)*
R.1.6	Podjęta próba uruchomienia kodu, udokumentowana zrzutem przedstawiającym uruchomiony program lub jego kompilację
R.1.7	Program nawiązuje zrozumiałą komunikację z użytkownikiem, wyświetlane komunikaty są znaczące *(jeżeli kod nie uruchamia się z powodu błędów kompilacji - sprawdzić w kodzie aplikacji)*
R.2	**Rezultat 2: Aplikacja konsolowa**
	Uwaga: kryteria 2.1 ÷ 2.7 należy sprawdzić w kodzie programu, sprawdzane elementy muszą być zapisane zgodnie ze składnią *Gdy aplikacja nie uruchamia się, a zdający zapisał zrzuty ekranu z uruchomienia aplikacji należy sprawdzić powód braku kompilacji. Jeśli występują błędy w plikach źródłowych zdającego kryteria 2.8, 2.9 nie są spełnione. Jeżeli błędy występują w innych plikach ocenić na podstawie kodu i zrzutu ekranu* *W przypadku języka Python argument self nie jest wliczany do liczby parametrów oraz deklaracja pól w konstruktorze przez self. np. self.pole*
R.2.1	Zdefiniowana klasa *Kosc* z polami publicznymi: dwa typu całkowitego, jedno logiczne oraz polem publicznym statycznym typu całkowitego
R.2.2	Nazwy obrazów od *kosc0.png* do *kosc6.png* zostały przypisane jako elementy tablicy lub innej kolekcji
R.2.3	Konstruktor bezparametrowy losuje liczbę i przypisuje ją do pola wartości liczby oczek i pola identyfikatora pliku. Konstruktor z jednym argumentem przypisuje liczbie oczek i identyfikatorowi pliku: - wartość argumentu, gdy argument jest z zakresu <1, 6> - 0, w przeciwnym wypadku *(w języku Python jeden konstruktor z domyślną wartością argumentu)*
R.2.4	W przynajmniej jednym konstruktorze jest inkrementowane pole statyczne oraz polu logicznemu przypisywana jest wartość true
R.2.5	Metoda realizująca rzut kością losuje wartość z zakresu <1,6> i przypisuje ją do pola wartości liczby oczek i pola identyfikatora pliku tylko, gdy pole dostępności ma wartość true

R.2.6	Metoda blokująca kość jest bezparametrowa, nie zwraca wartości i ustawia pole logiczne na false
R.2.7	Istnieje metoda zwracająca wartość wyrzuconą na kości w postaci słownej np. "jeden" *(w przypadku wartości spoza zakresu <1, 6> - dowolnie)*
R.2.8	Program uruchamia się w konsoli, co jest widoczne na zrzucie ekranu
R.2.9	Uruchomiony program inicjuje jeden obiekt konstruktorem bezargumentowym, drugi jednoargumentowym. Po każdej inicjacji wyświetlana jest liczba instancji. Dla każdego obiektu wyświetlana jest wartość wyrzucona kością w postaci liczbowej i napisowej oraz odpowiadająca jej nazwa pliku *(w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)*
R.3	**Rezultat 3: Aplikacja mobilna**
	Uwaga: należy uwzględnić różnice pomiędzy emulacjami - takie cechy jak marginesy, wielkości bloków itp. nie należy brać pod uwagę. Kryteria 3.1 ÷ 3.5 sprawdzić w kodzie źródłowym, sprawdzane elementy muszą być zapisane zgodnie ze składnią. *Gdy aplikacja nie uruchamia się, a zdający zapisał zrzuty ekranu z uruchomienia aplikacji należy sprawdzić powód braku kompilacji. Jeśli występują błędy w plikach źródłowych zdającego kryteria 3.6 ÷ 3.10 nie są spełnione. Jeżeli błędy występują w innych plikach lub bibliotekach sprawdzić w kodzie oraz na zrzucie ekranu. Dla Android Studio dopuszcza się także rozwiązanie w języku Kotlin*
R.3.1	Zastosowany język znaczników XML/XAML lub inny do opisu interfejsu użytkownika oraz zdefiniowano przynajmniej jedną kontrolkę
R.3.2	Zastosowane kolory tła: okna lub rozkładu: #ED27C121, przycisku: #ED275021, rozmiar czcionki pola tekstowego ustawiono na 40
R.3.3	Zastosowane marginesy zewnętrzne dla obrazów i przycisku 10, wyśrodkowano wszystkie elementy widoku
R.3.4	Dla przycisku oraz dla przynajmniej jednego obrazu zdefiniowana metoda kliknięcia, która jest powiązana z kontrolką
R.3.5	Program odwołuje się do przynajmniej jednej kontrolki w sposób zgodny z danym środowiskiem programistycznym
R.3.6	Aplikacja kompiluje się i uruchamia w emulatorze, co jest udokumentowane zrzutem ekranu, układ wszystkich kontrolek jest zgodny z obrazem 2 lub 3 w arkuszu egzaminacyjnym *(w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)*
R.3.7	W stanie początkowym aplikacji wszystkie obrazy wyświetlają grafikę *kosc0.png* bez przezroczystości, przycisk jest opisany „RZUT” oraz pole tekstowe zawiera napis 0 *(w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)*
R.3.8	Kliknięcie przycisku powoduje, że są losowane wartości tylko dla kostek dostępnych, odpowiadające wartościom obrazy są wyświetlane *(w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)*
R.3.9	Kliknięcie przycisku powoduje wyświetlenie pod nim sumy wszystkich oczek na wyświetlanych obrazach *(w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)*
R.3.10	Dla wszystkich kości: gdy dana kość jest wyświetlana bez przezroczystości, po kliknięciu widoczna jest przezroczystość. Po kolejnym kliknięciu, kość jest wyświetlana bez przezroczystości *(w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)*
R.4	**Rezultat 4: Testy aplikacji**
	Uwaga: Kryteria 4.1 ÷ 4.3 sprawdzić w kodzie źródłowym, sprawdzane elementy muszą być zapisane zgodnie ze składnią. Zrzuty ekranu z kryteriów 4.4 ÷ 4.7 muszą zawierać cały obszar ekranu z widocznym paskiem zadań. Dokumentacja z kryterium 4.8 zapisana jest w pliku egzamin
R.4.1	Co najmniej w jednej metodzie testującej została wykorzystana asercja, nazwy metod odzwierciedlają cel testu, zastosowane narzędzie testujące odpowiednie dla języka
R.4.2	W teście metody realizującej rzut kością sprawdzone czy wartość jest z zakresu <1,6>

R.4.3	W teście metody realizującej rzut kością sprawdzone czy wartość się nie zmienia jeżeli pole dostępność jest ustawione na false
R.4.4	Uruchomiona przynajmniej jedna metoda testująca udokumentowana zrzutem ekranu
R.4.5	Wynik uruchomionego testu z R.4.2 lub R.4.3 wynika z danych testowych co jest udokumentowane zrzutem ekranu
R.4.6	Istnieje przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji konsolowej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.7	Istnieje przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji mobilnej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.8	Dokumentacja zawiera: nazwę systemu operacyjnego, nazwy środowisk programistycznych, nazwy języków programowania, nazwy emulatora dla aplikacji mobilnej
R.4.9	Przygotowana jest dokumentacja w postaci płyty: oba projekty są spakowane oraz zmodyfikowane pliki są na zewnątrz archiwów