CENTRALNA KOMISJA EGZAMINACYJNA

Arkusz zawiera informacje prawnie chronione do momentu rozpoczęcia egzaminu

Nazwa kwalifikacji: **Projektowanie, programowanie i testowanie aplikacji**
Symbol kwalifikacji: **INF.04**
Numer zadania: **02**
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.04-02-26.01-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 aplikacji konsolowej oraz dokumentację aplikacji mobilnej zgodnie z opisem w części III instrukcji do zadania. Wykorzystaj konto Egzamin bez hasła.

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

Część I. Aplikacja konsolowa

Wykorzystując zasady programowania obiektowego zaprogramuj klasy do realizacji quizu o nazwach: *Pytanie* oraz *PytanieZamkniete*.

Założenia do programu:

- Program wykonywany w konsoli
- Zastosowany obiektowy język programowania zgodny z zainstalowanym na stanowisku egzaminacyjnym: C++ lub C#, lub Java, lub Python
- Program powinien podejmować zrozumiałą komunikację z użytkownikiem
- W programie może być zastosowane angielskie lub polskie nazewnictwo zmiennych i funkcji
- Program powinien być zapisany czytelnie, z zachowaniem zasad czystego formatowania kodu, należy stosować znaczące nazwy pól i metod
- Uruchomienie programu i sprawdzenie działania klas zgodnie z poleceniami w części III arkusza

Klasa *Pytanie*, której obiekt **nie może** być powołany, zawiera:

- Pola dostępne jedynie w klasie oraz w klasach potomnych, przechowujące:
  - Wartość tekstową z treścią pytania
  - Nazwę pliku zawierającego zdjęcie do pytania
  - Informację czy odpowiedź na pytanie jest poprawna, typu logicznego
- Konstruktor dwuargumentowy, którego argumentami są treść pytania i nazwa pliku dla pytania
  - Na podstawie argumentów ustawia wartości pól
  - Ustawia pole logiczne: odpowiedź na pytanie nie jest poprawna
- Metodę abstrakcyjną typu logicznego, sprawdzającą odpowiedź. Metoda ma jeden argument o typie znakowym, który przyjmuje wartości ‘A’ lub ‘B’ lub ‘C’ (odpowiedzi użytkownika na pytanie quizu). Metoda nie zawiera domyślnej implementacji

Klasa *PytanieZamkniete* dziedzicząca po klasie *Pytanie*. Klasa zawiera:

- Pola dostępne jedynie w klasie, niedostępne w klasach potomnych, przechowujące:
  - Treść odpowiedzi A
  - Treść odpowiedzi B
  - Treść odpowiedzi C
  - Informację o poprawnej odpowiedzi (znak ‘A’ lub ‘B’ lub ‘C’)
- Konstruktor sześcioargumentowy, o argumentach: treść pytania, nazwa pliku graficznego dla pytania, treść odpowiedzi A, treść odpowiedzi B, treść odpowiedzi C, odpowiedź poprawna
  - Konstruktor wywołuje konstruktor klasy, po której dziedziczy

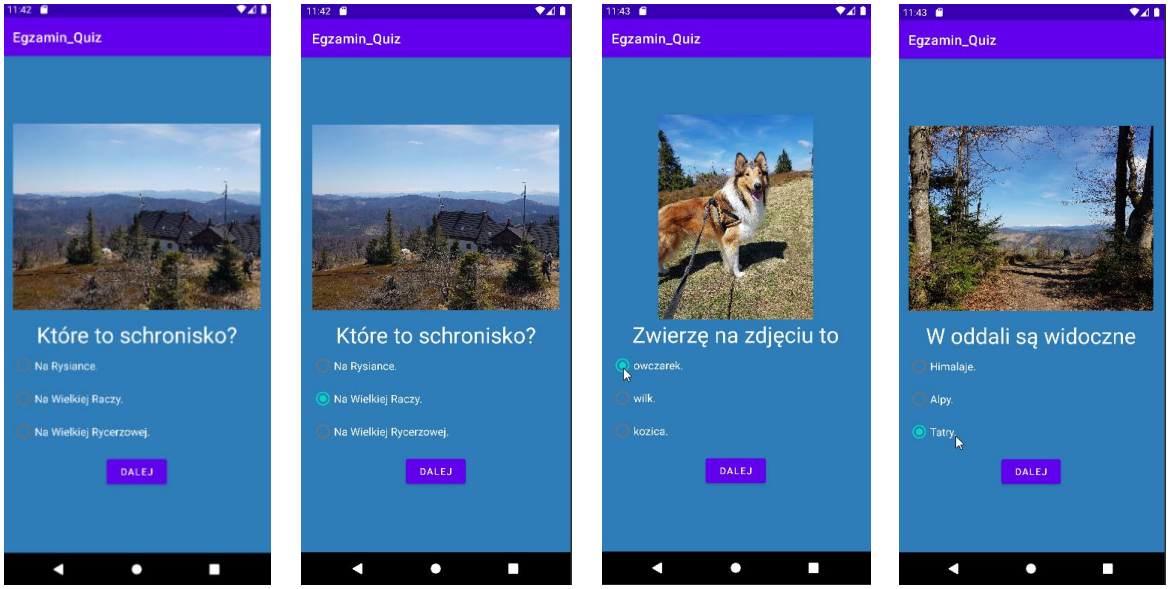
  - Przypisuje treści odpowiedzi oraz poprawną odpowiedź do pól klasy
- Implementację metody abstrakcyjnej sprawdzającej odpowiedź, która:
  - Sprawdza czy odpowiedź jest poprawna i w zależności od wyniku, przypisuje odpowiednią wartość do pola logicznego
  - Zwraca wartość pola logicznego

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 oraz plik wykonywalny, jeżeli istnieje.

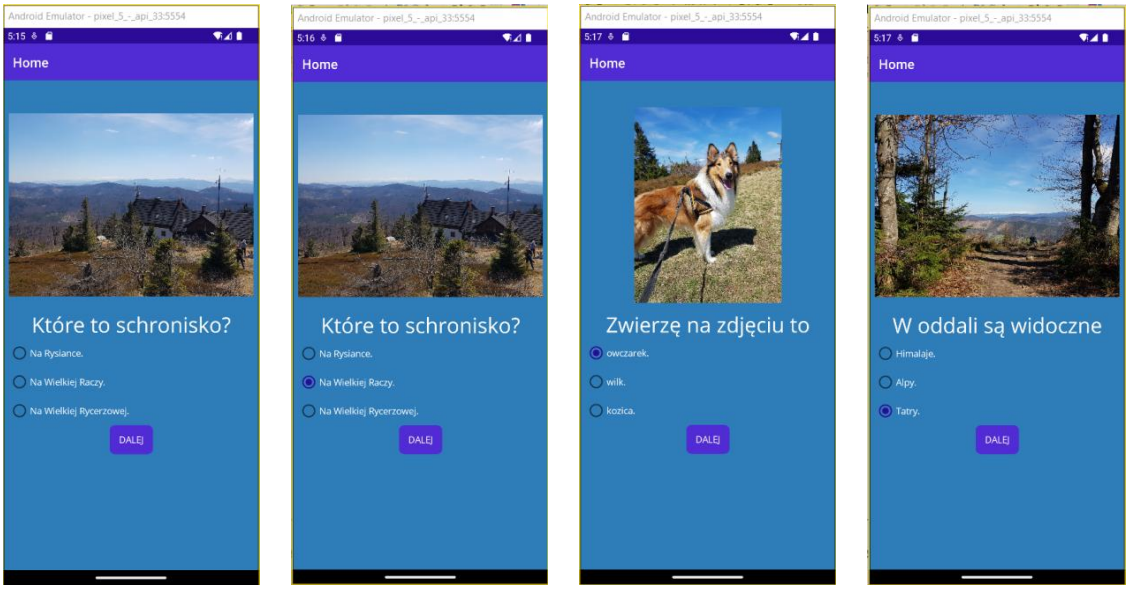
Część II. Aplikacja mobilna

Za pomocą środowiska programistycznego dostępnego na stanowisku egzaminacyjnym wykonaj aplikację mobilną quizu dotyczącego gór. Do zbudowania aplikacji zastosuj obrazy i treść pytań z archiwum *zad2.7z* zabezpieczonego hasłem **?Quiz5**

Podczas programowania aplikacji zmiany kodu rejestruj za pomocą repozytorium wersji Git. Po przygotowaniu projektu utwórz lokalne repozytorium wersji Git, podczas pracy utwórz co najmniej 2 zapisy wersji: po utworzeniu widoku aplikacji oraz po zapisaniu działania aplikacji. Komentarze migawek (*commit*) powinny być znaczące. Skonfiguruj repozytorium Git: jako nazwę użytkownika użyj numer zdającego, jako email użytkownika egzamin@poczta.pl



Obraz 1. Stany aplikacji mobilnej w Android Studio



Obraz 2. Stany aplikacji mobilnej w .NET MAUI lub XAMARIN

Elementy aplikacji z wartościami początkowymi:

- Obraz z wartością początkową *zad1.jpg*
- Pole tekstowe z treścią pytania: „Które to schronisko?”
- Trzy pola radio z odpowiedziami: „Na Rysiance.”, „Na Wielkiej Raczy.”, „Na Wielkiej Rycerzowej.”
- Przycisk DALEJ

Założenia dotyczące widoku:

- Interfejs użytkownika zapisany za pomocą języka znaczników wspieranego w danym środowisku (np. XAML, XML)
- Zastosowany dowolny rozkład pozwalający na rozmieszczenie elementów zgodnie z obrazami 1 lub 2
- Tło okna lub rozkładu #2E7CB8
- Biały kolor czcionki dla elementów
- Rozmiar czcionki z treścią pytania większy niż pozostałe treści
- Jednocześnie można wybrać tylko jedno pole radio
- Wyśrodkowane w poziomie obraz, treść pytania i przycisk
- Wyrównane do lewej strony pola typu radio

UWAGA: Po utworzeniu widoku aplikacji zapisz stan kodu za pomocą Git

Działanie aplikacji (można wykorzystać klasy *Pytanie, PytanieZamkniete* z aplikacji konsolowej):

- Treść pytań i odpowiedzi można skopiować z pliku *pytania.txt* wypakowanego z archiwum ZIP
- Wszystkie pytania należy zapisać w tablicy, liście lub innej kolekcji. Aplikacja jest uniwersalna - zawsze działa na wszystkich elementach kolekcji
- Po wybraniu przycisku DALEJ:
  - Inkrementowana jest liczba punktów, jeśli wybrano prawidłową odpowiedź
  - Jeżeli istnieją następne pytania w quizie, wyświetlane jest kolejne pytanie
  - W przeciwnym wypadku wyświetlane jest ponownie pytanie pierwsze
  - Wszystkie pola radio nie są zaznaczone
- Aplikacja powinna być zapisana czytelnie, z zachowaniem zasad czystego formatowania kodu, należy stosować znaczące nazwy zmiennych i funkcji

UWAGA: Po zaprogramowaniu aplikacji zapisz stan kodu za pomocą Git

Podejmij próbę kompilacji i emulacji aplikacji. Informacje dotyczące dokumentacji umieszczono w części III zadania.

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

Część III. Testy aplikacji i dokumentacja

Sprawdź działanie klas zaimplementowanych w I części egzaminu.
Zweryfikuj możliwość utworzenia obiektu klasy *Pytanie*. Po próbie uruchomienia programu wykonaj zrzut ekranu z otrzymanym błędem i zapisz pod nazwą *test1.png* w folderze *dokumentacja*. Następnie umieść ten kod w komentarzu.
W programie głównym zainicjuj obiekt klasy *PytanieZamkniete*, dane do konstruktora wczytaj z klawiatury. Wczytaj z klawiatury odpowiedź na pytanie. Sprawdź za pomocą metody poprawność odpowiedzi i wyświetl na ekranie tekst „Odpowiedź prawidłowa" lub „Odpowiedź nieprawidłowa". Wykonaj zrzut ekranu o nazwie *test2.png*.

Zrzuty zapisz w folderze *dokumentacja*. Na zrzutach należy umieścić okno z wynikiem działania programu oraz otwarte środowisko programistyczne lub okno terminala z kompilacją projektu. Jeżeli aplikacja nie uruchamia się z powodu błędów kompilacji, należy na zrzucie umieścić okno ze spisem błędów i widocznym otwartym środowiskiem programistycznym.

Wykonaj zrzuty ekranu dokumentujące działanie aplikacji mobilnej: *mobile1*, *mobile2* ... (np. stan początkowy, po wybraniu odpowiedzi i wciśnięciu przycisku, z kolejnym pytaniem). Zrzuty powinny zawierać cały obszar ekranu z widocznym paskiem zadań. Zrzuty zapisz w folderze *dokumentacja*

Wykorzystując polecenia Git lub wbudowane narzędzia systemu kontroli wersji w środowisku programistycznym wypisz wszystkie zapisane stany kodu (migawki) wraz z komentarzami oraz z konfiguracją użytkownika Git (email i nazwa). Wykonaj zrzut/zrzuty ekranu dokumentujące tą czynność. Zrzuty nazwij *git1.png*, *git2.png*, ... i zapisz w folderze *dokumentacja*

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

Zrzuty ekranu i dokument zapisz w folderze *dokumentacja*.

UWAGA: Nagraj płytę z rezultatami pracy. W folderze z numerem zdającego powinny się znajdować foldery: dokumentacja, konsolowa, mobilna. W folderze dokumentacja: pliki ze zrzutami, plik egzamin. W folderze konsolowa: spakowany cały projekt aplikacji konsolowej, pliki źródłowe, plik wykonywalny jeśli istnieje. W folderze mobilna: spakowany cały projekt aplikacji mobilnej, pliki z kodem źródłowym interfejsu i logiki. 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 i dokumentacja

Dokumentacja z poleceniami Git pobrana ze strony https://git-scm.com/book/en/v2 dostęp 6 lipca 2023
git config
Git has a default way of doing hundreds of things. For a lot of these things, you can tell Git to default to doing them a different way, or set your preferences. This involves everything from telling Git what your name is to specific terminal color preferences or what editor you use. There are several files this command will read from and write to so you can set values globally or down to specific repositories.
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com

git help
The git help command is used to show you all the documentation shipped with Git about any command.
git init
To take a directory and turn it into a new Git repository so you can start version controlling it, you can simply run git init.
git add
The git add command adds content from the working directory into the staging area (or “index”) for the next commit. When the git commit command is run, by default it only looks at this staging area, so git add is used to craft what exactly you would like your next commit snapshot to look like.
git status
The git status command will show you the different states of files in your working directory and staging area. Which files are modified and unstaged and which are staged but not yet committed. In its normal form, it also will show you some basic hints on how to move files between these stages.
git diff
The git diff command is used when you want to see differences between any two trees. This could be the difference between your working environment and your staging area (git diff by itself), between your staging area and your last commit (git diff --staged), or between two commits (git diff master branchB).
git commit
The git commit command takes all the file contents that have been staged with git add and records a new permanent snapshot in the database and then moves the branch pointer on the current branch up to it.
git log
The git log command is used to show the reachable recorded history of a project from the most recent commit snapshot backwards.

Wypełnia zdający

Do arkusza egzaminacyjnego dołączam płytę CD/DVD opisaną numerem

PESEL | | | | | | | | | | |**, której jakość nagrania została przeze mnie sprawdzona.**

Wypełnia Przewodniczący ZN

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

……
Czytelny podpis Przewodniczącego ZN