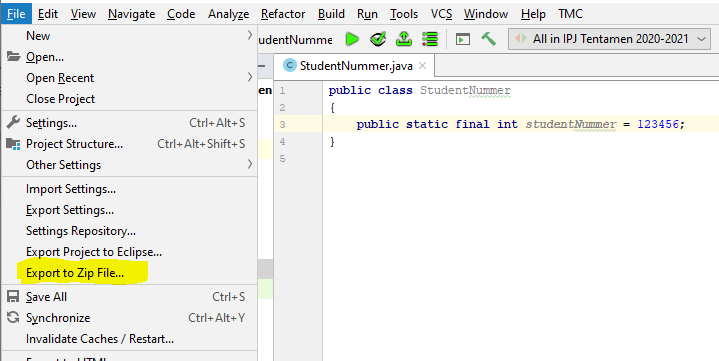
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Voorblad in te leveren door student? Zo ja ☒, dan volgende blok invullen | | | | | | | | | | | |
| Studentgegevens | | | | | | | | | | | |
| Naam student |  | | | | | | | | | | |
| Studentnummer |  | | | | Klas | | | |  | | |
| Kruis hiernaast de naam van de eigen vakdocent aan |  | | Edwin van der Geld | |  | Ik volgde geen les | | | | | |
|  | | Hans van der Linden | |  | Ik weet het niet | | | | | |
|  | | Johan Talboom | |  |  | | | | | |
| Vermeld op ieder blad je naam, studentnummer en klas | | | | | | | | | | | |
| Algemene gegevens van de toets | | | | | | | | | | | |
| Academie | | AE&I | | Fase | | | VT | | | Periode | 1.1 |
| Opleiding | | TI | | Onderwijsvorm | | | | VT | | | |
| Naam vak | | Introductie Programmeren in Java | | Vakcode | | | | EITI-IPJ | | | |
| Subtitel | | Oefen-praktijktoets | | Auteur(s) | | | | Hans van der Linden | | | |
| Datum | | *Geen (oefentoets)* | |  | | | | Johan Talboom | | | |
| Tijd | | *(100 min)* | | Reviewer(s) | | | | Maurice Snoeren | | | |
| Aantal pagina’s (inclusief voorblad) | |  | | Aantal opgaven | | | | 3 (ieder met deelopgaven) | | | |
| Zak/slaaggrens | | De cesuur is 55%. 55 van de 100 punten voor een voldoende. | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Bijzonderheden | | | |
| Papier (kruis aan wat van toepassing is) | Lijntjes A4 | Kladpapier | Ruitjes A4 |
|  |  | Anders nl: Antwoordformulier |
| Opgave inleveren? | (☒=ja □=nee) | | |
| Rekenmachine | Technische Informatica: géén rekenmachine toegestaan | | |
| Toegestane hulpmiddelen | Geen hulpmiddelen toegestaan | | |
| Boeken (titel, auteur) boek titel, auteur, druk | | |
| Anders nl: **WEL** toegestaan is gebruik van digitaal lesmateriaal op Brightspace, eigen laptop met ontwikkeltools, eigen aantekeningen, uitwerkingen van oefenopgaven en oefentoetsen, online naslagwerken over Java;  **NIET** toegestaan tijdens de toets is uitwisselen van bestanden met anderen, communiceren met anderen (email, chat), code letterlijk overnemen van internetbronnen. | | |
| Opmerkingen | Lees de instructie voor downloaden van de aangeleverde Java code en het LARP project aan het begin van de toets. De opgaven worden uitgereikt op papier. Je maakt de uitwerking op je laptop en levert die via Brightspace in. Let goed op de inleverinstructie aan het begin van de toets. | | |
| Contactgegevens academiebureau AE&I | Gebouw LA, kantoor LA003  Barbara Klijs: 088-5257494  Olaf van Maurik: 088-5259293  Gebouw CHL 13  Irene van Gaalen: 088-5257388 | | Vakdocent (naam en tel.nr.):  Johan Talboom 0885258644  Edwin van der Geld 0885253278  Hans van der Linden 0885257234 |
|  | | | |
| Elke student wordt geacht de bepalingen m.b.t. het afleggen van de toetsen te kennen. | | | |

**Inleverinstructie**

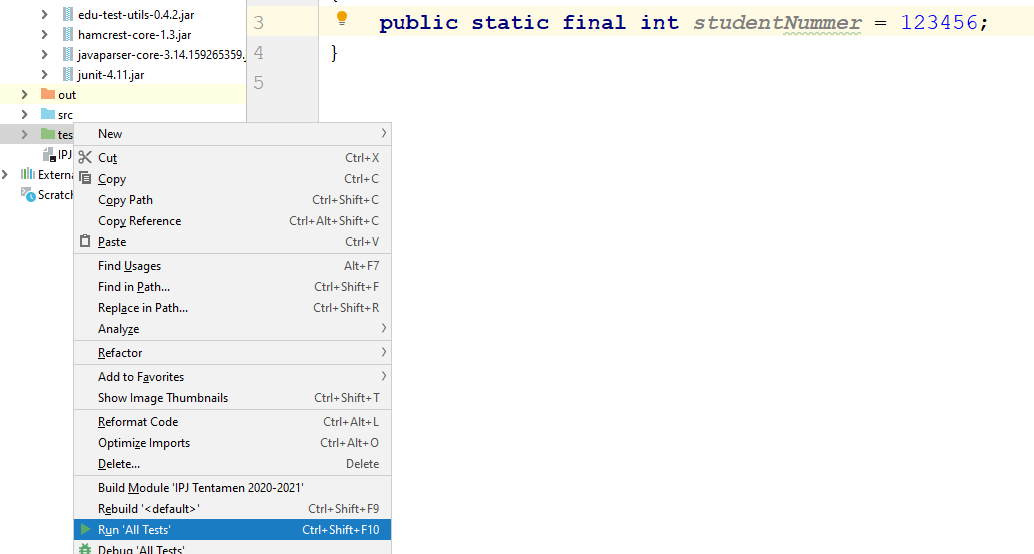
Je uitwerking van deze toets bestaat uit een **IntelliJ project** en een **LARP project**. Om die in te leveren maak je een zip-file van jouw project vanuit IntelliJ en kopieer je daarin ook het LARP project. De zip-file kan je maken via het menu: File 🡪 Export to Zip File… .



Daarna **kopieer je je LARP file in de zojuist gemaakte zip-file** en lever je die zip-file in via de inleveropdracht op Brightspace in de module Introductie Programmeren in Java. Zorg ervoor dat jouw project in de root directory van de zipfile staat (dus niet in een subdirectory). Als de zipfile geopend wordt moet direct de projectfile (.iml) te zien zijn, naast de src map.

Geef jouw zipfile de bestandsnaam ‘Praktijktoets IPJ – ‘ gevolgd door jouw naam, dus bijvoorbeeld ‘Praktijktoets IPJ – Johan Talboom.zip’.

*Wie LARP op zijn laptop niet kan uitvoeren mag gebruik maken van een andere software tool waarmee flowcharts gemaakt kunnen worden. Lever in dat geval alle flowcharts ook als afbeelding (bijvoorbeeld .png of .jpg) in, want de docenten beschikken wellicht niet over jouw software tool.*

 Dit tentamen kun je zelf testen door middel van de aangeleverde testcode (net zoals bij TMC). Je kunt echter niet de TMC-testknop gebruiken, maar moet de tests vanuit IntelliJ uitvoeren. Deze testen geven **NIET** aan wat er mis is, alle foutmeldingen zijn uit de testcode gehaald. Om de testcode uit te voeren, kun je in het project rechtsklikken op het mapje ‘*tests’*, en hierna op ‘*run all tests*’ klikken

**De casus voor deze toets**

Uit een lijst van songs willen we playlists kunnen samenstellen. De songs hebben als belangrijkste eigenschappen een lengte (in seconden) en een genre (muziekstijl). Een playlist is een volgorde waarin bepaalde songs worden afgespeeld. Een playlist kan songs van één genre bevatten of songs van alle mogelijke genres, en dat wordt bij het aanmaken van de Playlist bepaald. Een playlist kan ook een beperking in de totale lengte van alle songs in de playlist hebben.

**Succes!**

Opgaven 0 Wie ben je?

Begin met het downloaden van het project van Brightspace. Hier staan al een aantal klassen in.

Zet eerst je eigen studentnummer in de klasse StudentNummer. Deze klasse staat al in het project:

**public class** StudentNummer  
{  
 **public static final int *studentNummer*** = 123456;  
}

Let op dat je de juiste map van het project opent. Als je de verkeerde map in Intellij opent, zal het project niet compileren. Sommige code compileert nog niet door het ontbreken van de klassen die je nog moet maken.

## Opgave 1 De muziek (20 punten totaal)

1a De klasse Song 4 punten

Maak een klasse Song met de attributen

* title, hier slaan we de naam van het nummer in op
* length, hier slaan we in op hoe lang dit nummer is, in seconden
* genre, hier slaan we het genre in op, bijvoorbeeld “Hiphop”, “Jazz” of “Pop”

Bedenk hierbij wat de juiste datatypes zijn en gebruik die.

1b Constructor 4 punten

Maak een default constructor aan voor de klasse Song, die de attributen initialiseert, waarbij title en genre op een lege string gezet wordt en length op -1.

1c Tweede Constructor 4 punten

Maak een tweede constructor aan voor de klasse Song, waarbij waarden voor de 3 attributen worden meegegeven als parameters. Let hierbij op de volgorde, eerst title, dan length, dan genre.

1d Getters 4 punten

De klasse Song hoeft geen attributen aan te passen, maar er moeten wel attributen uitgelezen kunnen worden. Voeg getters toe voor title, length en genre en gebruik hiervoor de standaard Java naamgeving.

1e toString 4 punten

Voeg een toString-methode toe die de titel, lengte (in minuten/seconde notatie) en het genre teruggeeft.

## Opgave 2 Een playlist (40 punten totaal)

2a Klasse Playlist 10 punten

Maak de klasse Playlist, Deze klasse heeft de volgende eigenschappen

* atribuut songs die een lijst met songs opslaat
* attribuut requiredGenre, waar het genre van deze lijst in opgeslagen wordt
* attribuut maxLength, waar de maximale lengte van de playlist in seconden in wordt opgeslagen
* een constructor zonder parameters. Deze zet requiredGenre op de lege string, en maxLength op -1
* een constructor met een parameter voor requiredGenre. De maxLength word op -1 gezet
* een constructor met een parameter voor maxLength.

Let dat je alle attributen goed initialiseert.

2b Methode length 3 punten

Voeg de methode length toe aan de klasse Playlist. Deze methode geeft de totale lengte van deze playlist terug.

2c Getters 4 punten

Voeg getters toe voor de attributen requiredGenre en maxLength. Setters zijn in de playlist niet van toepassing.

2d methode getSong 3 punten

Voeg een methode getSong toe aan de klasse Playlist met een int parameter die de index van de song in de arraylist songs teruggeeft. Indien de index niet bestaat geeft de methode null terug.

2e methode genreIsRequired 3 punten

Voeg een private methode genreIsRequired toe aan de klasse Playlist met een boolean returntype. Deze methode geeft true terug als deze playlist een vast genre heeft, en false als deze playlist verschillende genres kan bevatten.

2f methode maxLengthIsSet 3 punten

Voeg een private methode maxLengthIsSet toe aan de klasse Playlist met een boolean returntype. Deze methode geeft true terug als deze playlist een vast maximale lengte heeft, en false als deze playlist geen maximale lengte heeft.

2g methode addSong 6 punten

Voeg de methode boolean addSong(Song newSong) toe aan de klasse Playlist die de song toevoegt, als dit toegestaan is (playlist mag niet te lang worden, en moet voldoen aan het genre). Als de song niet toegevoegd kan worden geeft de methode false terug, anders true

2h methode testAddSong 4 punten

Maak een static methode testAddSong() in de Main klasse aan die een song met een verkeerd genre aan een playlist voor één specifiek genre probeert toe te voegen en een foutmeldingstekst afdrukt op de console als dat toch lukt. Roep testAddSong() aan vanuit de main methode.

2i methode testAddSong 4 punten

Breid de methode testAddSong() van klasse Main uit zodat hij ook een song aan de playlist probeert toe te voegen die de maximum lengte van de Playlist zou overschrijden en een foutmeldingstekst afdrukt op de console als dat toch lukt (dus als er een fout in de code zit).

## Opgave 3 De speler (40 punten totaal)

3a klasse Player 6 punten

Maak een klasse Player met daarin

* Een attribuut allSongs, waar alle songs in worden opgeslagen
* Een attribuut genrePlaylists, een ArrayList waarin voor ieder genre een eigen Playlist word opgeslagen
* Een default constructor waar alle attributen geinitialiseerd worden

3b methode addSong 4 punten

Maak de methode addSong(Song song) in de klasse Player. Deze methode voegt een song toe aan het attribuut allSongs, maar alleen als deze nog niet voorkomt in de player.

3c methode genreExistsInPlaylists 5 punten

Maak in klasse Player een private methode boolean genreExistsInPlaylists(String genre) die true terug geeft als (minstens) één van de genrePlaylists van de Player het genre als required genre heeft, en false als het genre van geen van de genrePlaylists het required genre is.

3d methode addSongToPlaylist 5 punten

Maak in klasse Player een private void methode addSongToPlaylist die een Song meekrijgt als parameter. De methode voegt deze song toe aan die playlist uit genrePlaylists die het genre van song als required genre heeft.

3e distributeSongsOverPlaylists 10 punten

Ontwerp een flowchart voor een methode distributeSongsOverPlaylists() van klasse Player die alle songs van de player verdeelt over de genrePlaylists op basis van genre. Als voor het genre van een song nog geen playlists bestaat in de player, dan wordt er eentje aangemaakt voor dat genre en wordt de song daaraan toegevoegd. Als de playlist voor dat genre wel al bestaat wordt de song aan die playlist toegevoegd.

Er is een LARP project aangeleverd waarin al een MAIN module is gemaakt en een module om songs aan te maken. Er zijn ook al drie lege modules aangemaakt die je zelf gaat invullen. Twee daarvan zijn voor de methoden die je net in Java aan Player hebt toegevoegd.

***Let op: vanwege de beperkte tijd in deze toets moet de flowchart wel de structuur en de logische stappen van het algoritme tonen, maar hoeft hij niet in LARP uitvoerbaar te zijn.***

3f distributeSongsOverPlaylists 6 punten

Implementeer de methode distributeSongsOverPlaylists() (waarvoor je net een flowchart hebt gemaakt) in Java. De methode krijgt geen parameters en heeft void return type.

3g showAllPlaylists 4 punten

Maak voor Player een methode void showAllPlaylists() die alle playlists uit genrePlaylists afdrukt.

***Deze laatste opgave is extra oefening en hoort niet tot het oefententamen:***

3h createRandomPlaylist (extra oefening)

Maak voor de klasse Player een methode createRandomPlaylist(int maxLength) met als return type Playlist. Deze methode maakt een Playlist aan waarin songs van willekeurig gekozen genres worden toegevoegd zolang de totale length van die songs kleiner of gelijk is aan maxLength. Er mogen echter niet twee songs van hetzelfde genre na elkaar in de playlist voorkomen.

**[einde van de toets]**