Labs

(1) Level A	(1) Level B	(1) Level C
Zombie Dice	Connect Four Players	Video Store

Zombie Dice

Directions

Write a program that allows (2-5) players to alternate playing Zombie Dice until there is a winner. The game will start by getting players names and then generates a random turn order.

How the games works:

- The game consists of 13 dice
  - 6 Green Dice
    - Faces: 2xRunners, 3xBrains, 1xShots
  - 4 Yellow Dice
    - Faces: 2xRunners, 2xBrains, 2xShots
  - 3 Red Dice
    - Faces: 2xRunners, 1xBrains, 3xShots
- A player wins after she / he collects 13 brains and stops rolling.
- Turns alternate until there is a winner.
- Player turns
  - A play has the following choices each turn
    - Roll dice
      1. Gathering Dice
         a. Runners are picked up
         b. New dice are drawn until the player has 3 dice to roll or the zombie bucket is empty.
      2. Rolling Dice
         a. Rolled brains are held
         b. Rolled shots are held
         c. Runners will be rerolled if the player continues.
      3. Check for death
         a. If a player has 3 or more dice
            - His/her turn ends
            - All brains earned this turn are lost
    - Stop
      - All brains gathered in the turn added the players score
      - If the player has 13 or more brains he / she wins the game.

Coding

Main:

	Description
Methods	
main	Uses the ZombieDiceBucket and ZombieDie classes to simulate players playing a game of Zombie Dice. Required Data: String[] names – Stores player names int[] scores – Stores the player scores
shuffleNames(String[] names)	Shuffles the order of the items in the names array.
findWinner(String[] names, int[] scores)	Returns a String containing the name of the winning player or null if there is no winner yet.

ZombieDie (abstract class)

	Description
Constructors	
ZombieDie(int dieColor)	Sets the dieColor.
Public Static Final Attributes	
int NOT_ROLLED	Stores 0 and is used for rolling
int RUNNER	Stores 1 and is used for rolling
int BRAIN	Stores 2 and is used for rolling
int SHOT	Stores 3 and is used for rolling
int RED	Stores 1 and is used for die color
int GREEN	Stores 2 and is used for die color
int BLUE	Stores 3 and is used for die color
Attributes	
int dieColor	Stores the color of the die
Methods	
int getValue()	Returns the value
int getDieColor()	Returns the dieColor
void setDieColor(int dieColor)	Sets the dieColor
String toString()	Returns a text value of the die. • The text for un-rolled is its color. • The text for rolled text is its text-value
abstract void roll()	Randomly sets the value of the die based on its dieColor.

RedZombieDie extends ZombieDie

	Description
Constructors	
RedZombieDie()	Calls RedZombieDie constructor with the text "Red"
Methods	
void roll()	Randomly sets the value of the die based on its faces: • 2xRUNNER • 1xBRAIN • 3xSHOT

YellowZombieDie extends ZombieDie

	Description
Constructors	
YellowZombieDie()	Calls YellowZombieDie constructor with the text "Yellow"
Methods	
void roll()	Randomly sets the value of the die based on its faces: • 2xRUNNER • 2xBRAIN • 2xSHOT

GreenZombieDie extends ZombieDie

	Description
Constructors	
GreenZombieDie()	Calls GreenZombieDie constructor with the text "Green"
Methods	
void roll()	Randomly sets the value of the die based on its faces: • 2xRUNNER • 3xBRAIN • 1xSHOT

ZombieDiceBucket

	Description
Constructors	
ZomibeDiceBucket(String name, int color)	Creates the dice ArrayList.
Methods	
void loadBucket()	Clears dice and then loads 6 green dice, 4 yellow dice and 3 red dice.
ZombieDie draw()	Removes and returns 1 random die or null when the bucket is empty.

Grading:

Other Grades:

- ZombieDiceBucket Class

Minor Grades:

- Die Classes

2 Minor Grades

- Main

Connect Four Players

Directions

Modify Connect Four to allow two players to play the game. To build the new version of the game you will need to add the following classes: Player, ComputerPlayer and HumanPlayer.

Player will be an abstract class that can store either a ComputerPlayer or HumanPlayer. When the game starts the user will be asked what type of player they want as red and what type of player they want as black.

When a ComputerPlayer moves it will randomly select a valid move.

When a HumanPlayer moves he/she will be prompted for moves until a valid one is entered.

Coding

Main:

	Description
Methods	
Main	Uses the ConnectFourGame, Player, HumanPlayer, ComputerPlayer objects to making a working connect four game where the user picks the type of players.

Player (abstract class)

	Description
Constructors	
Player(String name, int color)	Sets the name and color.
Attributes	
String name	The name of the player
int color	Stores RED or BLACK
Methods	
String getName()	Returns the name of the player
int getColor()	Returns the color of the player
abstract int move(ConnectFourGame game)	Receives a ConnectFourGame. Returns a valid move. Note: This method does not change the game or it's board.

HumanPlayer extends Player

	Description
Constructors	
HumanPlayer(String name, int color)	Calls super to set name and color
Methods	
int move(ConnectFourGame game)	Receives a ConnectFourGame. Returns a valid move. This method asks the player for moves until a valid move is entered.

ComputerPlayer extends Player

	Description
Constructors	
ComputerPlayer(String name, int color)	Calls super to set name and color
Methods	
int move(ConnectFourGame game)	Receives a ConnectFourGame. Returns a valid move. This method generates random moves until a valid move is created.

Grading:

Other Grades:

- Player

Minor Grades:

- HumanPlayer
- ComputerPlayer

2 Minor Grades

- Main

Video Store

Directions

Create a program that will allow a user to rent games and/or movies from a video store. The user will have the following options when the program starts: rent movie, rent game and check out.

When either rent option is picked the user will be given a list of the titles in that category and an option to cancel. When an item is selected from the list it will print all the information for that title and give the user the option to either rent the item or cancel.

When the user selects check out it will print a receipt. The receipt will include the title of each rented item, each item's cost, the subtotal, tax and total.

Notes:

- When an item is rented it should no longer show up under the list of items that can be rented.
- Your store will need 4 games and 4 movies

Coding:

MainFile

	Description
Methods	
Main	Creates a VideoStore and loads it with 4 games and 4 videos. Once the VideoStore is created the user will shop in the store and put items into a rented games ArrayList and rented videos ArrayList When the user checks out they will be told what they rented, how much each item was to rent, the subtotal, the amount of tax they will pay and their total

Rental

	Description
Constructors	
Rental(String title, double cost, int rating)	Sets title, cost, rating to the received values and sets rented to false.
Attributes	
String title	The title of the item
double cost	The cost to rent this title
int rating	The rating category of the title
boolean rented	Stores if the title has been rented or not
Methods	
String getTitle()	Returns the title
double getCost()	Returns the cost
int getRating()	Returns the rating
boolean getRented()	Returns rented
void setRented(boolean rented)	Changes rented to the received value
String toString()	Returns a String containing the title, cost. Note: Rating will be taken care of by the subclasses.

Video extends Rental

	Description
Constructors	
Video(String title, double cost, int rating, String description, String director, ArrayList<String> leadingActors, int minutes)	Calls super to set title, cost, rating and rented. Sets description, director, leadingActors and minutes to the received values.
Public Static Final Attributes	
int G	Stores a 0 and is used for rating
int PG	Stores a 1 and is used for rating
int PG13	Stores a 2 and is used for rating
int R	Stores a 3 and is used for rating
Attributes	
String director	Stores the name of the movie director
String description	Stores the description of the movie
ArrayList<String> leadingActors	Stores the main actors in the movie
int minutes	Stores how long the movie is.
Methods	
String getDescription()	Returns the description
String getDirector()	Returns the director
ArrayList<String> getLeadingActors()	Returns the leading actors of the movie
int getMinutes()	Returns the minutes
String toString()	Returns a String that stores the information for the movie.

Game extends Rental

	Description
Constructors	
Game(String title, double cost, int rating, String platform, int numberOfPlayers)	Calls super to set title, cost, rating and rented. Sets platform and numberOfPlayers to the received values.
Public Static Final Attributes	
int G	Stores a 0 and is used for rating
int EVERYONE	Stores a 1 and is used for rating
int EVERONE_TEN_PLUS	Stores a 2 and is used for rating
int TEEN	Stores a 3 and is used for rating
int MATURE	Stores a 4 and is used for rating
int ADULTS ONLY	Stores a 5 and is used for rating
Int RATING PENDING	Stores a 6 and is used for rating
Attributes	
String platform	Stores the name of the system that can play the game
int numberOfPlayers	Stores the number of players
Methods	
String getPlatform()	Returns the platform
int getNumberOfPlayers()	Returns the number of players
String toString()	Returns a String that stores the information for the game.

VideoStore

	Description
Constructors	
VideoStore(ArrayList<Rental> rentals)	Sets rentals.
VideoStore()	Sets rentals to an empty ArrayList.
Attributes	
ArrayList<Video> rentals	Stores the videos in the store.
Methods	
ArrayList<Video> getVideos()	Returns the videos in the store.
ArrayList<Game> getGames()	Returns the games in the store.
void addRental(Rental r)	Adds the received rental.
int gamesInStock()	Returns the number of non-rented games
int videosInStock()	Returns the number of non-rented videos

Grading:

Other Grades:

- Rental
- VideoStore

Minor Grades:

- Video
- Game

2 Minor Grades:

- Main