Πρώτη Σειρά ασκήσεων
Ημερομηνία Παράδοσης: Δευτέρα 25 Μαΐου 3:00 μ.μ.

Για την άσκηση αυτή θα προγραμματίσετε σε Java το παιχνίδι blackjack (γνωστό και ως 21), όπως παίζεται σε ένα τραπέζι σε ένα καζίνο.

Κανόνες του παιχνιδιού
Στο blackjack ο στόχος είναι να πάρεις χαρτιά (χέρι) ώστε το σύνολο των πόντων τους να είναι όσο πιο κοντά στο 21 γίνεται (χωρίς να το ξεπερνάει), και να είναι περισσότεροι από αυτούς του dealer. Τα χαρτιά δίνουν τόσους πόντους όσο το νούμερο τους, οι φιγούρες δίνουν 10, και ο άσσος δίνει 1 ή 11 πόντους, ότι είναι καλύτερο για το σκορ του παίχτη. Το παιχνίδι παίζεται ως εξής. Ο παίχτης ποντάρει ένα ποσό (bet). Ο dealer έχει ένα ρεύμα χαρτιών (river) που αποτελείται από μία ή παραπάνω τράπουλες. Μοιράζει από ένα χαρτί σε όλους τους παίχτες και τραβάει ένα κι αυτός. Όλα τα χαρτιά είναι ανοιχτά. Στη συνέχεια μοιράζει άλλο ένα γύρο από χαρτιά σε όλους και ένα δικό του, κλειστό.
Αρχικά ο dealer τσεκάρει αν έχει blackjack (21 πόντους με δύο φύλλα). Αν έχει τότε παίρνει τα στοιχήματα από όλους τους παίχτες, εκτός από αυτούς που έχουν κι αυτοί blackjack. Αν δεν έχει, και κάποιος παίχτης έχει blackjack, πληρώνεται αυτόματα 1.5 φορές τα λεφτά που πόνταρε.
Στη συνέχεια ο dealer παίζει το χέρι του κάθε παίχτη. Αν ο παίχτης έχει δύο όμοια φύλλα (π.χ., δύο ρηγάδες) τότε μπορεί να κάνει split. Αυτό σημαίνει ότι ουσιαστικά έχει δύο χέρια το καθένα με το ένα από τα φύλλα. Στη συνέχεια παίζει κανονικά το κάθε χέρι. Η άλλη επιλογή που έχει ο παίχτης πριν ξεκινήσει να παίζει είναι να κάνει double, δηλαδή να διπλασιάσει το στοίχημα του. Αν διπλασιάσει, τότε ο dealer του δίνει **μόνο ένα** χαρτί και σταματάει. Αν δεν κάνει τίποτα από αυτά, παίζει κανονικά το χέρι του. Σε κάθε βήμα έχει την επιλογή να πάρει φύλλο (hit) ή να σταματήσει (stand). Παίρνει χαρτιά μέχρι να αποφασίσει να σταματήσει. Το τελικό σκορ του παίχτη είναι το άθροισμα των πόντων των χαρτιών του. Αν ξεπεράσει τους 21 πόντους χάνει αυτόματα (καίγεται), και πληρώνει το στοίχημα.
Ο dealer παίζει όταν τελειώσει με όλους τους παίχτες. Ο dealer τραβάει χαρτιά όσο το άθροισμα των πόντων του είναι λιγότερο από 17. Αν ξεπεράσει τα 21 καίγεται και πληρώνει όσους παίχτες έχουν μείνει (δεν έχουν καεί ή δεν έχουν blackjack). Αλλιώς κερδίζει αυτός που έχει το μεγαλύτερο άθροισμα πόντων, και ο νικητής πληρώνεται όσο είναι το στοίχημα. Αν έχουμε ισοπαλία στους πόντους δεν κερδίζει κανένας.
Μπορείτε να διαβάσετε τους κανόνες εδώ.

Υλοποίηση
Για την υλοποίηση σας θα ακολουθήσετε τα εξής βήματα.

Βήμα 1 (25%): Υλοποίηση του ρεύματος των χαρτιών για τον dealer. Το βήμα αυτό περιλαμβάνει την δημιουργία δύο κλάσεων:

Η κλάση **Card**: Κρατάει πληροφορία για ένα χαρτί. Για το παιχνίδι μας δεν μας νοιάζει η σειρά/το χρώμα του χαρτιού, αλλά μόνο το νούμερο/φιγούρα του. Η κλάση κρατάει το νούμερο/φιγούρα (String) που αρχικοποιείται στον constructor, και την αξία του χαρτιού που υπολογίζεται. Χρειαζόμαστε και τις εξής μεθόδους:

1. **getValue**: Επιστρέφει την αξία του χαρτιού. Ο άσσος έχει αξία 1 και οι φιγούρες 10.
2. **isAce**: Επιστρέφει boolean τιμή αν είναι άσσος ή όχι
3. **equals**: Η γνωστή μας μέθοδος που επιστρέφει true αν τα χαρτιά έχουν ίδια φιγούρα/νούμερο.
4. **toString**: Επιστρέφει απλά την φιγούρα/νούμερο.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Card.java**

Υπόδειξη: Χρησιμοποιείστε την κλάση Integer για να πάρετε την αριθμητική αξία ενός String που αναπαριστά έναν αριθμό.

Η κλάση **River**: Υλοποιεί το ρεύμα των χαρτιών. Η κλάση αυτή θα αρχικοποιείται με τον αριθμό από τράπουλες που έχει το ρεύμα. Έχει πεδίο ένα πίνακα με αντικείμενα Card που κρατάει όλα τα χαρτιά από τις τράπουλες (η σειρά δεν έχει σημασία), ο οποίος δημιουργείται και γεμίζει στον constructor. Θα έχει επίσης τις εξής μεθόδους:

5. **nextCard**: Αυτή είναι η βασική μέθοδος της κλάσης, η οποία επιστρέφει ένα τυχαίο χαρτί (Card) από τα εναπομείναντα χαρτιά του ρεύματος. Για την υλοποίηση χρειάζεστε ένα πεδίο cardsLeft που μας λέει πόσα χαρτιά έχουν μείνει στο ρεύμα. Δημιουργήστε μια τυχαία τιμή μεταξύ 0 και cardsLeft-1 (χρησιμοποιώντας την κλάση Random). Θα επιστρέψετε το χαρτί σε αυτή τη θέση. Το χαρτί δεν διαγράφεται, απλά θα το ανταλλάξετε (swap) με το χαρτί στην θέση cardsLeft-1 και θα μειώσετε το cardsLeft κατά 1. Αν δεν υπάρχουν χαρτιά επιστρέψτε null.
6. **shouldRestart**: Επιστρέφει true αν ο αριθμός των χαρτιών που έχουν μείνει πέσει κάτω από το ένα τέταρτο των αρχικών χαρτιών (θα είναι χρήσιμο να το κρατήσετε σε ένα πεδίο numberOfCards).
7. **restart**: Ανακατεύει ξανά τις τράπουλες. Αυτό γίνεται απλά θέτοντας το cardsLeft ίσο με το numberOfCards οπότε πλέον η τυχαία επιλογή γίνεται από όλα τα χαρτιά.

Αποθηκεύστε τον κώδικα σας στο αρχείο **River.java.**

Δημιουργήστε μια μέθοδο **main** στην κλάση River η οποία θα δημιουργεί ένα ρεύμα χαρτιών με μία τράπουλα. Μέσα σε ένα while loop τραβάτε χαρτιά και τα εκτυπώνετε μέχρι η shouldRestart επιστρέψει true. Κάνετε restart. Τραβάτε χαρτιά και εκτυπώνετε μέχρι το χαρτί που θα επιστραφεί να είναι null.

Υπόδειξη: Στην κλάση River θα σας βολέψει να ορίσετε ένα πίνακα-σταθερά από String με όλα τα νούμερα/φιγούρες για την δημιουργία των αντίστοιχων χαρτιών. Η σειρά δημιουργίας των χαρτιών δεν έχει σημασία

Βήμα 2 (15%): Υλοποίηση ενός χεριού.

Σε αυτό το βήμα θα υλοποιήσετε την κλάση **Hand** η οποία θα κρατάει πληροφορία για ένα χέρι. Η κλάση έχει ένα πεδίο ArrayList από χαρτιά (Card). Μπορείτε να ορίσετε και άλλα πεδία αν τα χρειάζεστε. Έχει επίσης τις εξής μεθόδους:

1. **addCard**: Παίρνει σαν όρισμα ένα χαρτί και το προσθέτει στο χέρι.
2. **score**: Υπολογίζει τους πόντους του χεριού. Αν το χέρι δεν περιέχει άσσο, τότε είναι απλά το άθροισμα των αξιών του χεριού. Αν το χέρι περιέχει τουλάχιστον ένα άσσο, και το άθροισμα των τιμών του χεριού είναι P, και P+10 ≤ 21, τότε οι πραγματικοί πόντοι του χεριού είναι P+10, γιατί ο άσσος αποτιμάται για 11 πόντους.
3. **canSplit**: Επιστρέφει μια Boolean τιμή αν το χέρι μπορεί να γίνει split, δηλαδή περιέχει δύο ίδια χαρτιά.
4. **split:** Καλείται όταν θέλουμε να κάνουμε split ένα χέρι και επιστρέφει ένα πίνακα με δύο νέα χέρια, στα οποία έχουμε προσθέσει από ένα από τα αρχικά χαρτιά του χεριού.
5. **isBlackjack:** Επιστρέφει μια Boolean τιμή αν το χέρι είναι blackjack, δηλαδή 21 με δύο φύλλα.
6. **isBust:** Επιστρέφει μια Boolean τιμή αν το χέρι έχει «καεί» ή όχι.
7. **toString:** Επιστρέφει ένα String με τα χαρτιά χωρισμένα με κενό.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Hand.java.**

Δημιουργήστε μια μέθοδο **main** στην κλάση Hand. Μέσα στην main δημιουργήστε ένα χέρι στο οποίο θα προσθέσετε δύο χαρτιά, που θα είναι και τα δύο άσσοι. Τυπώστε το χέρι, το σκορ του, και το αποτέλεσμα της canSplit. Κάνετε split το χέρι και αποθηκεύσετε το αποτέλεσμα. Τυπώστε τα δύο χέρια. Στο πρώτο χέρι προσθέσετε ένα ρήγα. Εκτυπώστε το χέρι, το σκορ του χεριού, και το αποτέλεσμα της isBlackjack. Προσθέστε άλλο ένα άσσο. Εκτυπώστε το χέρι, και το σκορ του χεριού. Προσθέστε ένα δεκάρι. Εκτυπώστε το χέρι, το σκορ του χεριού, και το αποτέλεσμα της isBust.

Προαιρετικά, μπορεί η toString να επιστρέφει και το σκορ του χεριού. Αν εχει άσσο και μπορεί να μετρήσει και ως 1 και ως 10, (π.χ., 4 Α) θα πρέπει να επιστρέφει 5/15.

Βήμα 3 (25%): Υλοποίηση ενός παίχτη. Το βήμα αυτό περιλαμβάνει την δημιουργία δύο κλάσεων:

Η κλάση **CasinoCustomer** η οποία θα κρατάει πληροφορίες για ένα θαμώνα του καζίνο. Η κλάση έχει πεδία το όνομα του παίκτη, και τα χρήματα τα οποία έχει για να παίξει. Τα πεδία αυτά αρχικοποιούνται στον constructor. Η κλάση διαχειρίζεται ότι έχει να κάνει με την χρηματική κατάσταση του παίχτη. Θα έχει τις εξής μεθόδους:

1. **payBet**: Παίρνει σαν όρισμα το ποσό ενός πονταρίσματος το οποίο έχασε ο παίχτης και το αφαιρεί από τα χρήματα του παίχτη.
2. **collectBet**: Παίρνει σαν όρισμα το ποσό ενός πονταρίσματος το οποίο κέρδισε ο παίχτης και το προσθέτει στα χρήματα του παίχτη.
3. **canCover**: Παίρνει σαν όρισμα το ποσό ενός πονταρίσματος και επιστρέφει μια Boolean τιμή αν ο παίχτης έχει αρκετά χρήματα να το καλύψει.
4. **isBroke:** Επιστρέφει μια Boolean τιμή αν τα χρήματα του παίχτη πέσουν κάτω από το 1 ευρώ.
5. **toString:** Επιστρέφει ένα String με το όνομα του παίχτη.
6. **printState:** Τυπώνει το όνομα και τα χρήματα του παίχτη.

Αποθηκεύστε τον κώδικα σας στο αρχείο **CasinoCustomer.java**

Η κλάση **Player** η οποία θα κρατάει πληροφορίες για ένα παίχτη που παίζει στο τραπέζι για ένα γύρο. Η κλάση έχει πεδία ένα αντικείμενο CasinoCustomer, ένα αντικείμενο Hand, και μια πραγματική τιμή bet που είναι το στοίχημα του παίκτη. Η κλάση διαχειρίζεται το παιχνίδι του πελάτη για ένα γύρο. Θα έχει τις εξής μεθόδους:

1. Έναν constructor που παίρνει σαν όρισμα μόνο τον πελάτη και αρχικοποιεί το πεδίο.
2. Υπερφορτώστε τον constructor ώστε να παίρνει ορίσματα και να αρχικοποιεί και τα τρία πεδία.
3. Ορίστε μεθόδους **πρόσβασης** (accessor methods) για όλα τα πεδία.
4. **wins**: Η μέθοδος που καλείται όταν κερδίσει ο παίχτης. Τυπώνει ένα μήνυμα και ο πελάτης παίρνει τα χρήματα του στοιχήματος.
5. **winsBlackJack**: Η μέθοδος που καλείται όταν κερδίσει ο παίχτης με blackjack. Τυπώνει ένα μήνυμα και ο πελάτης παίρνει μιάμιση φορά τα χρήματα του στοιχήματος.
6. **loses**: Η μέθοδος που καλείται όταν χάσει ο παίχτης. Τυπώνει ένα μήνυμα και ο πελάτης πληρώνει τα χρήματα του στοιχήματος.
7. **placeBet**: Τυπώνει την οικονομική κατάσταση του παίχτη και του ζητάει να ποντάρει. Ελέγχει αν ο παίχτης έχει τα χρήματα και αν το στοίχημα είναι αποδεκτό (τουλάχιστον 1 ευρώ). Συνεχίζει να ρωτάει μέχρι να πάρει αποδεκτό στοίχημα.
8. **doubleBet**: Διπλασιάζει το στοίχημα.
9. **wantsToDouble**: Ελέγχει αν ο παίχτης έχει αρκετά χρήματα για να καλύψει το διπλάσιο στοίχημα, και αν ναι, ρωτάει τον χρήστη αν θέλει να διπλασιάσει το στοίχημα. Επιστρέφει μια boolean τιμή αν το χρήστης μπορεί και θέλει να κάνει double.
10. **wantsToSplit**: Ελέγχει αν ο παίχτης έχει αρκετά χρήματα για να καλύψει το διπλάσιο στοίχημα. Αν ναι, ρωτάει τον χρήστη αν θέλει να κάνει split. Επιστρέφει μια boolean τιμή αν το χρήστης μπορεί και θέλει να κάνει split.
11. **toString:** Επιστρέφει ένα String με το όνομα του παίχτη και το χέρι του.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Player.java**

Δημιουργήστε μια μέθοδο **main** στην κλάση Player. Δημιουργήστε ένα αντικείμενο CasinoCustomer με το όνομα σας και το ποσό των 50 ευρώ. Δημιουργήστε ένα αντικείμενο Player με τον πελάτη που δημιουργήσατε. Καλέστε την placeBet. Καλέστε τις wantsToSplit, και wantsToDouble. Καλέστε την wins, winsBlackjack και loses. Μετά από κάθε κλήση τυπώστε την οικονομική κατάσταση του παίχτη χρησιμοποιώντας την accessor method και την printState.

Βήμα 4 (10%): Υλοποίηση του dealer.

Σε αυτό το βήμα θα υλοποιήσετε την κλάση **Dealer** η οποία θα κρατάει πληροφορία για τον dealer του τραπεζιού για ένα γύρο. Η κλάση έχει πεδία ένα αντικείμενο River και ένα Hand. Το River αρχικοποιείται στον constructor. Έχει επίσης τις εξής μεθόδους:

1. Μεθόδο πρόσβασης (accessor method) για χέρι.
2. **draw**: Ο dealer τραβάει ένα χαρτί από το ρεύμα και το προσθέτει στο χέρι του.
3. **deal**: Παίρνει σαν όρισμα ένα αντικείμενο Player και προσθέτει ένα χαρτί από το ρεύμα στο χέρι του παίχτη.
4. **play**: Υλοποιεί το παιχνίδι του dealer. Όσο το σκορ του χεριού είναι λιγότερο από 17, συνεχίζει να τραβάει χαρτιά
5. **settle**: «Ταχτοποιεί» τις εκκρεμότητες του dealer με ένα παίχτη. Παίρνει σαν όρισμα ένα αντικείμενο Player και ελέγχει ποιο χέρι κερδίζει, αν κερδίζει κάποιο. Ανάλογα αν κέρδισε ή έχασε ο παίχτης πληρώνεται ή πληρώνει το στοίχημα (καλείται η wins/loses).
6. **toString:** Επιστρέφει ένα String “Dealer:” και το χέρι.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Dealer.java**

Δημιουργήστε μια μέθοδο **main** στην κλάση Dealer. Δημιουργήσετε ένα αντικείμενο River με μια τράπουλα, ένα αντικείμενο Dealer με το River που δημιουργήσατε. Καλέσετε την play και τυπώστε τον Dealer.
Αν έχετε υλοποιήσει τις κλάσεις CasinoCustomer και Player δημιουργήστε ένα Player με CasinoCustomer με το όνομα σας, και ένα χρηματικό ποσό που θα αποφασίσετε. Κάνετε deal 2 χαρτιά στον παίχτη, τυπώστε τον παίχτη, και καλέσετε την settle.

Βήμα 5 (15%): Υλοποίηση ενός γύρου του παιχνιδιού.

Σε αυτό το βήμα θα υλοποιήσετε την κλάση **Round** η οποία υλοποιεί ένα γύρο από το παιχνίδι. Έχει πεδία ένα αντικείμενο Dealer, και ένα ArrayList από Player που είναι οι παίχτες του γύρου. Επίσης έχει και μια ακόμη ArrayList από αντικείμενα Player που κρατάει τους παίχτες για τους οποίους ο dealer έχει εκκρεμότητα (πρέπει να κάνει settle). Ο constructor παίρνει σαν όρισμα ένα αντικείμενο River και δημιουργεί το αντικείμενο Dealer. Επίσης έχουμε τις εξής μεθόδους:

1. **addPlayer**: Παίρνει σαν όρισμα ένα αντικείμενο CasinoCustomer και δημιουργεί και προσθέτει τον αντίστοιχο Player στους παίχτες του τραπεζιού.
2. **playRound**: Υλοποιεί το βασικό παιχνίδι. Αρχικά, όλοι οι παίχτες ποντάρουν. Μετά ο dealer μοιράζει από ένα χαρτί σε κάθε παίχτη, και τραβάει και έναν γι αυτόν. Τυπώνουμε το χέρι του dealer ώστε να δούμε το φανερό χαρτί του. Στη συνέχεια ο dealer μοιράζει ένα δεύτερο χαρτί σε κάθε παίχτη και τραβάει ένα και γι αυτόν. Τυπώνουμε τα χέρια των παιχτών. Ελέγχουμε αν ο dealer έχει blackjack. Αν έχει μαζεύει τα λεφτά από τους παίχτες που δεν έχουν blackjack και ο γύρος τελειώνει.
   Αν δεν έχει blackjack ο dealer εξετάζει κάθε παίχτη ξεχωριστά. Αν ο παίχτης έχει blackjack πληρώνεται 1.5 φορές το στοίχημά. Αν όχι θα πρέπει να παίξει με τον παίχτη. Για το σκοπό αυτό βολεύει να κάνουμε βοηθητικές μεθόδους. Συγκεκριμένα, καλούμε την μέθοδο playPlayer παρακάτω. Όταν τελειώσει με τους παίχτες παίζει αυτός και ταχτοποιεί τις εκκρεμότητες (settle) με τους παίχτες που δεν έχουν καεί ή δεν έχουν κερδίσει με blackjack.
3. **Βοηθητικές μέθοδοι**: Για την υλοποίηση του παιχνιδιού με έναν παίχτη σας προτείνεται να κάνετε βοηθητικές μεθόδους (private) που θα χειρίζονται τις διάφορες περιπτώσεις. Μπορείτε να κάνετε και άλλη υλοποίηση αν προτιμάτε. Οι προτεινόμενες μέθοδοι είναι οι εξής:
   a. **playNormalHand**: Υλοποιεί την κλασσική περίπτωση που ο παίχτης τραβάει μέχρι να αποφασίσει να σταματήσει. Αν καεί χάνει και πληρώνει το στοίχημα, αλλιώς μπαίνει στην λίστα των παιχτών για να ταχτοποιηθεί (settle) αργότερα. Τυπώστε το χέρι κάθε φορά που προσθέτετε χαρτί.
   b. **playDoubledHand:** Υλοποιεί το παιχνίδι όταν ο παίχτης κάνει double.

c. **playSplitHand:** Υλοποιεί την περίπτωση που ο παίχτης κάνει split. Δημιουργεί δύο νέα αντικείμενα Player για καθένα από τα νέα χέρια (με το ίδιο στοίχημά και τον ίδιο πελάτη) και καλεί την playNormalHand για το καθένα.
d. **playPlayer**: Ελέγχει αν το χέρι είναι υποψήφιο για split και ο παίχτης θέλει και μπορεί να κάνει split ή αν θέλει και μπορεί να κάνει double, ή δεν ισχύει τίποτα από τα παραπάνω, και ανάλογα καλεί την αντίστοιχη από τις βοηθητικές μεθόδους.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Round.java**

Δημιουργήστε μια μέθοδο **main** στην κλάση Round. Δημιουργήστε ένα αντικείμενο River με 6 τράπουλες, και ένα αντικείμενο Round με αυτό το River. Δημιουργήστε και προσθέστε ένα CasinoCustomer με το όνομα σας και ποσό 100 ευρώ. Καλέσετε την playRound.

Βήμα 6 (10%): Υλοποίηση του τραπεζιού και του παιχνιδιού. Για το βήμα αυτό θα υλοποιήσετε δύο κλάσεις:

Την κλάση **BlackjackTable** η οποία υλοποιεί ένα τραπέζι blackjack. Έχει πεδία ένα αντικείμενο River, ένα πίνακα από CasinoCustomer αντικείμενα και τον αριθμό των συμμετεχόντων. Για το River θα χρησιμοποιούμε 6 τράπουλες. Ο αριθμός των συμμετεχόντων δίνεται σαν όρισμα στον constructor ο οποίος δημιουργεί και γεμίζει τον πίνακα. Θα ορίσετε και τις εξής μεθόδους:

1. **createCasinoCustomer**: Αυτή είναι βοηθητική μέθοδος (private) που ζητάει από τον χρήστη να δώσει για κάθε παίχτη το όνομα του και τα διαθέσιμα χρήματα που έχει δημιουργεί το αντικείμενο και το επιστρέφει. Θα την χρησιμοποιήσετε στον constructor.
2. **play**: Υλοποιεί το παιχνίδι. Σε κάθε γύρο δημιουργείτε ένα αντικείμενο Round στο οποίο προσθέτετε μόνο τους παίχτες που έχουν αρκετά χρήματα. Πριν ξεκινήσει ο γύρος ελέγχετε αν η ροή των χαρτιών πρέπει να ανανεωθεί, και αν ναι, κάνετε restart. Το παιχνίδι συνεχίζεται μέχρι όλοι οι παίχτες να χάσουν όλα τα χρήματα τους.

Μπορείτε να προσθέσετε και άλλες επιλογές αν θέλετε όπως για παράδειγμα να ρωτάτε τον κάθε παίχτη αν θέλει να συνεχίσει να παίζει. Ή να επιτρέπετε την προσθήκη νέου χρήματος σε ένα παίχτη (πρέπει να έχετε αντίστοιχη μέθοδο στην κλάση CasinoCustomer).

Την κλάση **Blackjack** η οποία υλοποιεί το παιχνίδι και έχει την **main**. Ρωτήστε τον χρήστη πόσους παίχτες θέλει να προσθέσει. Δημιουργήστε ένα αντικείμενο BlackjackTable και καλέστε την play.

Στο Παράρτημα στο τέλος της άσκησης σας δίνονται μερικά παραδείγματα εκτέλεσης του παιχνιδιού. Δεν είναι ανάγκη να έχετε ακριβώς το ίδιο output, ο στόχος είναι να σας δώσει μια κατεύθυνση.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- Δεν επιτρέπεται η χρήση public πεδίων στην άσκηση. Επίσης ο κώδικας θα πρέπει να είναι σωστά στοιχισμένος και καλά γραμμένος. Θα αφαιρεθούν βαθμοί από προγράμματα που είναι πολύ κακά γραμμένα.
- Θα τεστάρουμε και θα βαθμολογήσουμε την κάθε κλάση ξεχωριστά. Γι αυτό και θα πρέπει να σώσετε την κάθε κλάση σε ξεχωριστό αρχείο και να έχετε υλοποιήσει και την main. Θα πρέπει επίσης να κρατήσετε τα ονόματα και τα ορίσματα των public μεθόδων ακριβώς όπως σας ζητούνται για να μπορούμε να τρέξουμε την δική μας main. Θα αφαιρεθούν βαθμοί αν δεν το έχετε κάνει.
- Μια κλάση που δεν κάνει compile μηδενίζεται αυτόματα.
- Κάντε turnin τα προγράμματα σας στο assignment1@myy205.

  π.χ. turnin assignment1@myy205 Blackjack.java

  Στον κώδικα να αναγράφονται σε σχόλια το όνομα και ο ΑΜ σας.

Παράρτημα

Παράδειγμα 1:

```
C:[..]\assignment1>java Blackjack
Give the number of players:2
Give customer name and available money
Alice 100
Give customer name and available money
Bob 200

---- New Round! ---

Alice has 100.0 left
Alice place your bet: 20

Bob has 200.0 left
Bob place your bet: 40

Dealer: K
Alice: Q Q
Bob: 8 7

Player Alice: Q Q
Do you want to split: y
Alice: Q
Hit?: y
Alice: Q 6
Hit? n
Alice: Q
Hit?: y
Alice: Q 7
Hit? n

Player Bob: 8 7
Do you want to double?: n
Hit?: y
Bob: 8 7 9
Player Bob lost! Pay $40.0

Dealer: K 7
Player Alice lost! Pay $20.0
Tie with Alice. Nobody wins

---- New Round! ---

Alice has 80.0 left
Alice place your bet: 40

Bob has 160.0 left
Bob place your bet: 50

Dealer: 7
Alice: 9 2
Bob: 6 3
```

```
Player Alice: 9 2
Do you want to double?: y
Alice: 9 2 K

Player Bob: 6 3
Do you want to double?: y
Bob: 6 3 10

Dealer: 7 4 K
Tie with Alice. Nobody wins
Player Bob lost! Pay $100.0
```

Παράδειγμα 2:

```
C:[…]\assignment1>java Blackjack
Give the number of players:2
Give customer name and available money
Alice 100
Give customer name and available money
Bob 100

---- New Round! ---

Alice has 100.0 left
Alice place your bet: 50

Bob has 100.0 left
Bob place your bet: 20

Dealer: K

Alice: A Q
Bob: J 4

Blackjack! Player Alice collects $75.0

Player Bob: J 4
Do you want to double?: n
Hit?: y
Bob: J 4 7
Hit? n

Dealer: K 2 6
Player Bob won! Collect $20.0

---- New Round! ---

Alice has 175.0 left
Alice place your bet: 75

Bob has 120.0 left
Bob place your bet: 50

Dealer: K

Alice: J Q
Bob: 7 K

Player Alice: J Q
Do you want to double?: n
Hit?: n

Player Bob: 7 K
Do you want to double?: n
Hit?: n

Dealer: K 4 2 8
Player Alice won! Collect $75.0
```

```
Player Bob won! Collect $50.0

---- New Round! ---

Alice has 250.0 left
Alice place your bet: 100

Bob has 170.0 left
Bob place your bet: 100

Dealer: K

Alice: 5 9
Bob: 10 K
Dealer: K A
Player Alice lost! Pay $100.0
Player Bob lost! Pay $100.0

---- New Round! ---

Alice has 150.0 left
Alice place your bet: 100

Bob has 70.0 left
Bob place your bet: 70

Dealer: 3

Alice: 6 10
Bob: 4 6

Player Alice: 6 10
Hit?: n

Player Bob: 4 6
Hit?: y
Bob: 4 6 9
Hit? n

Dealer: 3 A 3
Player Alice lost! Pay $100.0
Player Bob won! Collect $70.0

---- New Round! ---

Alice has 50.0 left
Alice place your bet: 50

Bob has 140.0 left
Bob place your bet: 100

Dealer: 3

Alice: 5 4
Bob: 6 8
```

```
Player Alice: 5 4
Hit?: y
Alice: 5 4 6
Hit? n

Player Bob: 6 8
Hit?: y
Bob: 6 8 5
Hit? n

Dealer: 3 A 6
Player Alice lost! Pay $50.0
Player Bob lost! Pay $100.0

---- New Round! ---

Bob has 40.0 left
Bob place your bet: 40

Dealer: 8

Bob: J 5

Player Bob: J 5
Hit?: n

Dealer: 8 J
Player Bob lost! Pay $40.0
```