Object Oriented Programming 2020/21

Math in Casinos: Blackjack

MEEC – IST

1 Game

Blackjack is a popular American casino game, now found throughout the world. It is a banking game in which the aim of the player is to achieve a hand whose points total nearer to 21 than the banker's hand, but without exceeding 21.

Blackjack is played with an international 52-card deck without jokers. Casinos normally use several decks mixed together, called a **shoe**, both in order to speed up the game (with more cards in play you don't have to reshuffle after every single hand) and to make card counting more difficult (card counting is a technique used to gain advantage over the casino by keeping track of the proportions of different value cards remaining in play). The number of decks differs from casino to casino, but there can be anything from two to eight decks in the *shoe*. The reshuffling frequency also differs from casino to casino, and of course depends on the number of decks used. Frequent reshuffling is another way to diminish the value of card counting.

In casinos, blackjack is played on a specially designed table. There is a permanent dealer employed by the casino, and room for up to eight simultaneous players, playing against the dealer. Each player has a designated playing area in front of him where cards are placed and bets are made. The player needs chips to bet with. The standard denominations for casino chips are:

- white chips = \$1;
- red chips = \$5;
- green chips = \$25;
- black chips = \$100.

The betting limits should be clearly posted on a sign on the blackjack table. Normally, some of the most important rules, such as "Blackjack pays 3 to 2" and "Dealer must draw on 16 and stand on all 17's" are printed on the table.

Casino de Lisboa and Casino do Estoril use eight decks. They both pay 3 to 2 on a blackjack and stand on all 17's.

1.1 Betting and winning

Each player at the blackjack table has a circle or box to place bets in. There will always be a minimum bet and a maximum bet for the table. The maximum bet is normally ten to twenty times the minimum bet, meaning that a table with a $5 minimum would have a $50 to $100 maximum.

Each player decides how much to bet on a hand before the deal. Each hand will result in one of the following events for the player:

- **lose** – the player's bet is taken by the dealer.
- **win** – the player wins as much as he bet. If you bet $10, you win $10 from the dealer (plus you keep your original bet, of course).
- **blackjack (natural 21)** – the player wins 1.5 times the bet in a "Blackjack pays 3 to 2". With a bet of $10, you keep your $10 and win a further $15 from the dealer.
- **push** – the hand is a draw. The player keeps his bet, neither winning nor losing money.

1.2 Game objective

In blackjack, players don't play against each other and they don't co-operate. The only competition is the dealer.

The aim of the game is to accumulate a higher point total than the dealer, but without going over 21. You compute your score by adding the values of your individual cards. The cards 2 through 10 have their face value, J, Q, and K are worth 10 points each, and the ace is worth either 1 or 11 points (player's choice at the moment, with the value that assists his hand best). If the player holds an ace valued as 11, the hand is called **soft**, meaning that the player cannot go over 21 (**bust**) by taking an additional card; 11 plus the value of any other card will always be less than or equal to 21. Otherwise, the hand is **hard**. For instance, (ace, ace) is a soft 12 (2 or 12), (ace, ace, 8) is a soft 20 (10 or 20), (ace, ace, 4) is a soft 16 (6 or 16), (ace, ace, 10) is an hard 12 and (ace, ace, 4, 8) is an hard 14.

1.3 The deal and blackjack

At the start of a blackjack game, the players and the dealer receive two cards each. The players' cards are normally dealt face up, while the dealer has one face down, called the **hole card**, and one face up.

The best possible blackjack hand is an opening deal of an ace with any ten-point card. This is called a **blackjack**, or a **natural 21**, and the player holding this automatically wins unless the dealer also has a *blackjack*. If a player and the dealer each have a *blackjack*, the result is a *push* for that player. If the dealer has a *blackjack*, all players not holding a *blackjack* lose.

1.4 The players' turns

After the cards have been dealt, the game goes on with each player taking action – in clockwise order starting to dealer's left. A player must declare if he wants to take advantage of the **side rules** (explained below). You can use the side rules only once, when it's your turn to act after the deal. Then the player can:

- **stand** – keep his hand as it is.
- **hit** – take more cards from the deck, one at a time, until either the player judges that his hand is strong enough to go up against the dealer's hand and *stands*, or until it goes over 21, in which case the player immediately loses (**bust**).

In most places, players can take as many cards as they like, as long as they don't *bust*, but some casinos have restrictions regarding this.

1.5 The dealer's turn

When all players have finished their actions, either decided to *stand* or *busted*, the dealer turns over his hidden hole card. If the dealer has a *blackjack* with his two cards, he won't take any more cards. All players lose, except players who also have a *blackjack*, in which case it is a *push* – the bet is returned to the player.

If the dealer doesn't have a *blackjack*, he *hits* (takes more cards) or *stands*, depending on the value of the hand. Contrary to the player, though, the dealer's action is completely dictated by the rules. The dealer must *hit* if the value of the hand is lower than 17, otherwise the dealer will *stand*. Whether or not the dealer must *hit* on a soft 17 (a hand of 17 containing an ace being counted as 11) differs from casino to casino. There might even be blackjack tables with different rules within the same casino.

1.6 Showdown

If the dealer goes *bust*, all players who are left in the game *win*. Otherwise, if the dealer doesn't go bust, players with higher point totals than the dealer *win*, while players with lower totals than the dealer *lose*. For those with the same total as the dealer the result is a *push* – their stake is returned to them and they neither *win* nor *lose*.

Players with a *blackjack* win a bet plus a bonus amount, which is normally equal to half their original wager (in the case of "Blackjack pays 3 to 2"). A *blackjack* hand beats any other hand, also those with a total value of 21 but with more cards. As described above, if the dealer has a *blackjack*, players with *blackjack* make a *push*, while all other players *lose*.

1.7 Blackjack side rules

Numerous side rules allow for more intricate betting strategies. These side rules can only be used immediately after the deal, before you take any more cards. You cannot, for example, take a third card and then decide to double down. The most widely practiced side rules are explained below.

Insurance

When the dealer's face-up card is an ace, each player gets the chance to bet on whether the dealer has a blackjack. This is done before any other player actions.

The insurance wager equals your original bet and is used to cancel out the likely loss of this bet. A winning insurance bet will be paid at odds of 2:1, and since you lose your original bet, you'll break even on the hand. Strategy guides tend to advice against taking insurance.

Surrender

If you have a bad hand compared to the dealer's hand (judging from what you can see of it), you can give up the hand and reclaim half your bet. The casino keeps the other half uncontested. You need a really bad hand match-up for a surrender to be profitable, such as 16 vs the dealer showing a 10.

At some casinos, surrenders will not be allowed if the dealer has a *blackjack* (which he then checks for immediately after the deal). If the dealer has a *blackjack*, no surrenders will be granted and you'll lose the entire bet – unless you also have a *blackjack*, in which case it's a push. This side rule variation is called **late surrender**.

Splitting

When you get two starting cards of the same face value, you have the option to split the hand in two. You place another bet of the same size as the original bet and play on with two hands. It is legal to split 10-point cards even if they do not form a pair; for example, you could split a jack and a king. When you've decided to split a hand, the dealer immediately deals a second card to each hand. Now, if you're dealt yet another pair, some casinos allow you to split the hand again, while others don't.

When you're done splitting, each of your hands will be treated separately, meaning that you will take cards to your first hand until you *stand* or *bust*, and then carry on with the next hand.

If you split aces, you are dealt a second card to each hand as usual, but you are not allowed to take any further cards (unless you are dealt another ace and split again). All hands resulting from splitting aces remain as two-card hands.

If the second card dealt to a split ace is a 10-point card you do not receive the blackjack bonus for this hand. It does however win against an ordinary 21 made of more than two cards. If the dealer also has a *blackjack* the result for this hand is a push as usual. In many places the same rule (no blackjack bonus) is played if an ace is dealt as the second card to a 10-point card after splitting.

Doubling down

If you're fairly sure that your hand will beat the dealer's, you can double your original bet. You're sometimes allowed to double down for any amount up to the original bet amount. In most casinos you may double down on any hand, but some casinos require an opening hand worth 11, 10 or 9 (this is the case of Casino de Lisboa and Casino do Estoril).

When you've chosen to double down, you'll get one (and only one) more card from the dealer.

2 Card counting

John Ferguson (born 1943) known by his pen name, Stanford Wong, began playing blackjack in 1964 while teaching finance courses at San Francisco State University and getting his PhD in Finance from Stanford University in California. Not content with the teaching life, Wong agreed to be paid a salary of $1 for his last term of teaching at the university in order to not attend faculty meetings and to pursue his gambling career.

The term *wong* or *wonging* has come to mean a specific advantage technique in blackjack,

which Wong made popular in the 1980s. It involves watching the play of cards in a game without actually wagering your own money, until the count becomes advantageous, and then stepping in and playing only while the count remains in the player's favour, and then stepping out again. *Wonging* is the reason that some casinos have signs on some blackjack tables saying "No Mid-Shoe Entry", meaning that a new player must wait until exactly the first hand after a shuffle to begin playing. Wong is known to have been the principal operator of a team of advantage players that targeted casino tournaments including Blackjack, Craps and Video Poker in around Las Vegas.

The only reason the dealer has an advantage over the player is that he tests for a *bust* after the player; if the player busts the dealer wins immediately. As noted by Wong, this advantage can be reversed with card counting techniques.

2.1 Basic strategy

Dealer's advantage can be mitigated around to 1.005 win to the dealer using the best strategy for each of the player's hand and looking for the dealer's card according to Figure 1. Three tables are presented:

- **hard** – hands without aces, or where all aces value 1, and without an opening deal of a pair of aces.
- **soft** – hands with aces where at least one ace values 11 and without an opening deal of a pair of aces.
- **pairs** – both cards in the opening deal are of the same value.

The *pairs* hand table is only used once to decide if it is worth to split (once per possible split). Afterwards, the decision is ruled by the *hard* or *soft* hand tables, depending on the player's hand.

2.2 Hi-lo strategy

Note that the basic strategy assumes no memory on what has happened to the *shoe*. By counting cards, one can have an idea on the cards that are going to be dealt and increase significantly the odds for the player. A brief explanation of how to use the Hi-Lo strategy follows.

Step 1: Assign a point value to each rank, as presented in Table 1.

Step 2: Start with a **running count** of zero at the start of the deck/*shoe*. As cards are revealed, keep adding or subtracting from the *running count*, according to the point system in Step 1. For example, if the first ten cards to come out of the *shoe* were 3, 5, K, 7, Q, A, 8, 5, 4, 2, then the *running count* would be 1 +1 -1 +0 -1 -1 +0 +1 +1 +1 = +2.

Step 3: Divide the *running count* by the number of decks remaining, to get what is known as the *true count*. For instance, the *running count* is +7 and there are about 4 decks left. The *true count* would be $7/4 = 1.75$.

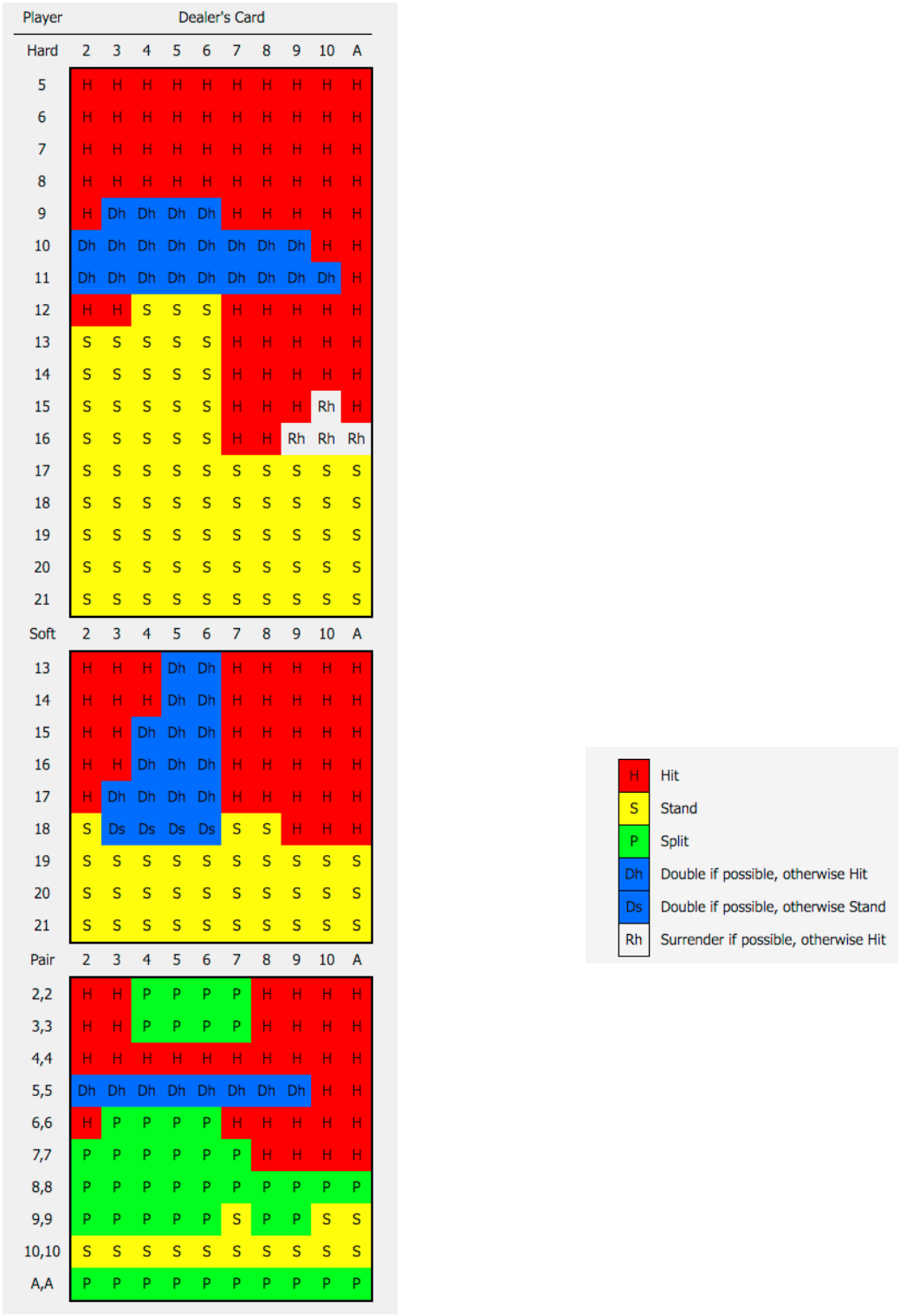

Figure 1: Basic strategy for a show with four or more decks.

RANK	VALUE
2	+1
3	+1
4	+1
5	+1
6	+1
7	0
8	0
9	0
10	-1
J	-1
Q	-1
K	-1
A	-1

Table 1: Hi-lo value.

Step 4: For some hands, you will play according to the *true count* and a table of **index numbers**, rather than basic strategy. The greater the count, the more inclined you will be to *stand*, *double*, *split*, take *insurance*, and *surrender*. The following rules are known as the **Illustrious 18** and **Fab 4**, respectively. Therein, the symbol "T" means ten.

Illustrious 18:

- Insurance: Insure at +3 or higher.
- 16vT: Stand at 0 or higher, hit otherwise.
- 15vT: Stand at +4 or higher, hit otherwise.
- TTv5: Split at +5 or higher, stand otherwise.
- TTv6: Split at +4 or higher, stand otherwise.
- 10vT: Double at +4 or higher, hit otherwise.
- 12v3: Stand at +2 or higher, hit otherwise.
- 12v2: Stand at +3 or higher, hit otherwise.
- 11vA: Double at +1 or higher, hit otherwise.
- 9v2: Double at +1 or higher, hit otherwise.
- 10vA: Double at +4 or higher, hit otherwise.
- 9v7: Double at +3 or higher, hit otherwise.
- 16v9: Stand at +5 or higher, hit otherwise.
- 13v2: Stand at -1 or higher, hit otherwise.
- 12v4: Stand at 0 or higher, hit otherwise.
- 12v5: Stand at -2 or higher, hit otherwise.

- 12v6: Stand at -1 or higher, hit otherwise.
- 13v3: Stand at -2 or higher, hit otherwise.

For example, the *index number* for a player 15 against a dealer 10 (15vT) is +4. This means the player should *stand* if the *true count* is +4 or higher, otherwise *hit*.

Fab4:

- 14vT: Surrender at +3 or higher, basic strategy otherwise.
- 15vT: Surrender at 0 or higher, basic strategy otherwise.
- 15v9: Surrender at +2 or higher, basic strategy otherwise.
- 15vA: Surrender at +1 or higher, basic strategy otherwise.

In the case of a 15vT, *Illustrious 18* and *Fab4* overlap. In this case, assume that if the *true count* is between 0 and +3 the player should *surrender*. If the *true count* is +4 or higher, the player should *stand*. Otherwise, the player should *hit*.

As a side remark note that in practice, in a Casino, Step 3 requires only a rough estimate. That is, for the example given therein, one can round 1.75 up to 2, to keep it simple. Moreover, the greater the *true count*, the more one should bet. This is where card counting becomes more art than science. Casino managers that know that players are counting might get mad with them! It helps avoid heat to keep the ratio of maximum bet to minimum bet to a limit, known as the **bet spread**. Only increasing bets after a win, only decreasing after a loss, and staying the same after a push, makes play look more natural, but at a cost to profitability.

2.3 Ace-five strategy

The ace-five strategy is the simplest counting strategy, and only advises how to bet according to the number of aces and fives that have been accounted for. The strategy is as follows:

1. Establish what your minimum and maximum bets will be. Usually the maximum will be 8, 16, or 32 times the minimum bet, or any power of 2, but you can use whatever *bet spread* you wish.
2. At the beginning of each *shoe*, start with your minimum bet, and a count of zero.
3. For each five observed, add one to the count.
4. For each ace observed, subtract one from the count.
5. If the count is greater than or equal to two, then double your last bet, up to your maximum bet.
6. If the count is less than or equal to one, then make the minimum bet.
7. Use basic strategy (with or without the *hi-lo* variation) for all playing decisions.

3 Implementation details

The following sections provide further details about project implementation, namely, minimum requirements, program parameters, commands and results, and running in the command line.

3.1 Minimum requirements

In this project it is required an implementation of the blackjack game as described in Section 1, with card counting techniques as described in Section 2. The implementation should include:

- one player only, besides the dealer;
- chips: white ($1), red ($5), green ($25), black ($100);
- there is a hole card (Section 1.3);
- no restriction in the number of cards taken as long as it does not bust (Section 1.4);
- the dealer must stand in all 17's (Section 1.5);
- blackjack pays 3 to 2 (Section 1.6);
- side rules (Section 1.7):
  - insurance (pays 2 to 1);
  - surrender (no late surrender);
  - splitting (allow resplitting until the player has as many as four hands and doubling a hand after a splitting);
  - doubling down (only on an opening hand worth 9, 10 or 11, and always doubles the bet; take only one more card from the dealer).
- card count techniques (basic, hi-lo and ace-five strategies).

3.2 Program parameters

There are three different modes for playing blackjack. An **interactive mode**, where the player is playing with the dealer through commands in the command line. A **debug mode**, where the game is fully loaded from a file. A **simulation mode** where the game is automatically played with a card counting strategy to understand the average gain in the balance of the player.

3.2.1 Input parameters for interactive mode

When running interactively commands from the command line the program should receive the following input parameters:

`min-bet`	minimum bet (`min-bet` $\geq$ \$1)
`max-bet`	maximum bet ($10 \times$ `min-bet` $\leq$ `maxbet` $\leq 20 \times$ `min-bet`)
`balance`	initial amount of money (`balance` $\geq 50 \times$ `min-bet`)
`shoe`	number of 52-card decks (without jokers) in the shoe ($4 \leq$ `shoe` ≤ 8)
`shuffle`	percentage of shoe played before shuffling ($10 \leq$ `shuffle` ≤ 100)

In the beginning of the game the cards are always shuffled. If `shuffle` is 60 it means that when 60% of the cards in the shoe are played then the cards are reshuffled again, and so on. After shuffling counts in the card counting strategies should be reseted!

3.2.2 Input parameters for debug mode

When loading the shoe and the commands from a file for debug purposes the program may receive as input parameters:

`min-bet`	minimum bet (`min-bet` $\geq$ \$1)
`max-bet`	maximum bet (10 $\times$ `min-bet` $\leq$ `maxbet` $\leq$ 20 $\times$ `min-bet`)
`balance`	initial amount of money (`balance` $\geq$ 50 $\times$ `min-bet`)
`shoe-file`	name of the file with the shoe (check examples of files in the Project webpage)
`cmd-file`	name of the file with the commands (check examples of files in the Project webpage)

In this case the shoe is loaded from a file and so it should not be shuffled in the beginning or at any other time. A few examples of a `shoe-file` and a `cmd-file` will be provided in the "Project section" of the OOP website. Stay tuned!

3.2.3 Input parameters for simulation mode

When performing a simulation the program may receive as input parameters:

`min-bet`	minimum bet (`min-bet` $\geq$ \$1)
`max-bet`	maximum bet (10 $\times$ `min-bet` $\leq$ `maxbet` $\leq$ 20 $\times$ `min-bet`)
`balance`	initial amount of money (`balance` $\geq$ 50 $\times$ `min-bet`)
`shoe`	number of 52-card decks (without jokers) in the shoe (4 $\leq$ `shoe` $\leq$ 8)
`shuffle`	percentage of shoe played before shuffling (10 $\leq$ `shuffle` $\leq$ 100)
`s-number`	number of shuffles to perform until ending the simulation
`strategy`	counting cards' strategy to use (combinations of `BS`, `HL` and `AF`)

The `shoe` parameter indicates the number of 52-card decks (without jokers) to use in the simulation and, in this case, the shoe must be shuffled in the beginning. Moreover, as for the interactive mode, after shuffling counts in the card counting strategies should be reseted!

Concerning the counting cards' strategy, the following combinations can be used: (i) `BS`; (ii) `BS-AF`; (iii) `HL`; (iv) `HL-AF`. In cases (i) and (iii) where `AF` is not used, then the following betting strategy should be used. Start with the minimum bet. If the player wins, increase the bet by `min-bet` (up to `max-bet`). If the player loses, decrease the bet by `min-bet` (down to `min-bet`). If the player pushes, keep the bet. Henceforward, this is called the **standard bet strategy**.

3.3 Commands and results

In the interactive mode, after entering the input parameters and loading the shoe the game should wait for player's commands. The possible commands are:

Command	Meaning
`b`	bet
`$`	current balance
`d`	deal
`h`	hit
`s`	stand
`i`	insurance
`u`	surrender
`p`	splitting
`2`	double
`ad`	advice
`st`	statistics

The player must decide how much to bet on an hand before the deal. So, the bet (`b`) command can be used as: `b` or `b 5`. If only `b` is typed then: (i) the previous betted amount is used; or (ii) the `min-bet` is used, if there was no previous bet. A `b` command typed after the deal and before the end of the dealer's turn is illegal, and so it should be printed in the terminal ``b: illegal command''. Other commands might be illegal in certain points of the game; in that case, just print a similar warning.

The current balanced (`$`) can be use at any time of the game. The deal (`d`) can only be used in the very beginning of the shoe or after the end of the dealer's turn, after the bet command. Immediately after the deal the player can use the side rules `i`, `u`, `p`, `2` (see Section 1.7 and Section 3.1). Examples of the results/prints of these commands are provided in the Project webpage.

Afterwards, hit (`h`) and stand (`s`) commands can be used. The `h` command prints to the terminal the actual hand of the player. For example, consider that the players hand is 2 and 3, and then hit is issued and a 4 is taken from the deck. In this case it should be printed to the terminal ``player's hand: 2 3 4''. The `s` command prints nothing, relative to the player, to the terminal. However, the dealer plays afterwards and so it should be printed to the terminal the dealers' actions (see examples provided in the Project webpage).

Each time the player wins/loses/pushes it should be printed in the terminal ``player WLP and his current balance is CB'', where `WLP` is win/lose/push, depending on the result, and `CB` is the current balance of the player. If the player and/or the dealer hold a blackjack then, before the previous message, it should be printed for the terminal ``blackjack!!'' followed by a new line.

When the cards in the shoe are shuffled print in the terminal ``shuffling the shoe...''. After shuffling counts in the card counting strategies should be reseted!

There is also two additional commands available: advice (`ad`) and statistics (`st`). The advice prints the next action the player should take, according to the basic, hi-lo and ace-five strategies (see examples provided in the Project webpage). The `st` command prints to the terminal the average statistics of the game per hand, with the following format:

```
BJ P/D     N1/N2
Win          N3
Lose         N4
Push         N5
Balance   N6(N7%)
```

where `N1/N2` is the average number of player/dealer's blackjacks per hand, and `N3`, `N4` and `N5` are the average number of times the player wins, loses and pushes per hand, respectively. The `N6` indicates the final balance of the player and `N7` the percentage of gain relative to the initial balance.

In debug mode the commands are read from the `cmd-file`. All commands described can appear in the `cmd-file` (including `ad` even if it is not considered/used!). In addition, the results/prints described for each command should also be printed in the terminal.

If the program is running in simulation mode (and only in this case), in the final of simulation a table with statistics (as previously detailed for the interactive mode) must be printed to the terminal. This is the only information required to be printed; that is, do not print the result of each command in the terminal. The commands to use in this case are completely dictated by the `strategy` received as input parameter.

3.4 Running in the command line

A .jar file must be created so that the program runs by typing in the terminal

```
java -jar <<YOUR-JAR-NAME>>.jar -i min-bet max-bet balance shoe shuffle
```

for the interactive mode, or by typing in the terminal

```
java -jar <<YOUR-JAR-NAME>>.jar -d min-bet max-bet balance shoe-file cmf-file
```

for the debug mode, or by typing in the terminal

```
java -jar <<YOUR-JAR-NAME>>.jar -s min-bet max-bet balance shoe shuffle s-number strategy
```

for the simulation mode.

If a Swing GUI is also provided (bonus point, see Section 3.1) the program should run by simply typing

```
java -jar <<YOUR-JAR-NAME>>.jar -g
```

in the terminal, and options should be chosen within the GUI.

4 Grading

The assessment will be based on the following 20-point scale:

1. **(5 point)**: UML. The UML will be evaluated, in a 5-point scale as: 0–very bad, 1.5–bad, 3–average, 4–good and 5–excellent.
2. **(15 points)**: A Java solution that provides an extensible and reusable framework. The implementation of the requested features in Java is also an important evaluation criteria. The following is pre-established on a 15-point scale:
   (a) **(8 points):** Correct implementation of the simulator: project visualization (2 points), interactive mode (2 points), debug mode (1 points), and simulation mode (3 points).
   (b) **(1 points):** Interfaces and polymorphism used correctly.
   (c) **(1 points):** Open-closed principle used correctly.
   (d) **(1 points):** `Object` class used correctly.
   (e) **(1 points):** Correct use of data structures (`Collection`, etc).
   (f) **(1.5 points):** Complete documentation of the simulator generated by the `javadoc` tool.
   (g) **(1.5 points):** Final report with a critical evaluation of the proposed solution.

Project visualization is conducted with a few examples of train and test sets (not provided by the professor previously). Note that the train and test sets are not at a specific folder in your machines (but rather in the client/professor machines, right?) so be careful about that. If the main parameteres are not correctly read from the terminal, visualization will not be possible and so 2-points are lost. See Section 3.4 for correct use of input parameters.

Finally, on a 20-point scale the following discounts apply:

1. **(-1 points):** Prints outside the format requested in Section 3.3.

2. **(-1 points):** A non-executable jar file, or a jar file without sources or with sources out of date. Problems in extracting/building a jar file, as well as compiling/running the executable in Java, both from the command line. Problems with Java versions; the Java executable should run properly in any PC/laptop.

3. **(-1 points):** Files submitted outside of the required format (see Section 5).

4. Projects submitted after the established date will have the following penalty: for each day of delay, there will be a penalty of 2^{n-1} points of the grade, where n is the number of days in delay. That is, reports submitted up to 1 day late will be penalized in $2^0 = 1$ points, incurring in a penalty of 1 point of the final grade; reports submitted up to 2 days late will be penalized in $2^1 = 2$ points of the final grade; and so on. Per day of delay we mean cycles of 24h from the day, hour and minutes specified for submission.

5 Deadlines and material for submission

The **deadline for submitting the project is May 24, before 14:00**. The submission is done via fenix, so ensure that you are registered in a project group.

The following files must be submitted:

1. An UML specification including classes and packages (as detailed as possible), in `.pdf` or `.jpg` format. Place the UML files inside a folder named UML.

2. An executable `.jar` (with the respective source files `.java`, compiled classes `.class`, and `MANIFEST.MF` correctly organized into directories).

3. Documentation (generated by the javadoc tool) of the application. Place the documentation inside a folder named JDOC.

4. A final report (up to 10 pages, in .pdf format) containing information that complements the documentation generated by the javadoc tool. Place the final report inside a folder named DOCS.

5. A self assessment form (in `.pdf` format) that will be made available in due time in the course webpage. Place the self assessment form inside the folder named DOCS (the same folder as the final report).

The UML folder, executable (the `.jar` file with the source files, besides the compiled files and `MANIFEST.MF`), the JDOC folder and the DOCS folder, should be **submitted via fenix in a single `.zip` file**.

The final discussion will be held from May 25 to June 2 (and if needed from June 7 to 9). The distribution of the groups for final discussion will be available in due time. All group members must be present during the discussion. **The final grade of the project will depend on this discussion, and it will be not necessarily the same for all group members.**