Hand

- hand[20]: Card

// Member Functions
+~Hand():
+ setHand(&d:Deck, i:int): void
+getHand(x:int):Card

Bet

- bet: float
- amount: float
-*lost: float

+ Bet():
+ Bet(b: float, a: float):
+ setBet(b:float): void
+ setAmount(a:float): void
+ getBet():float
+ getAmount():float
+ getLost(index:int):float
+ PlaceBets(p*:Player, numOfP:int):void

Deck

- arrCards[52]:Card
- index:int

+ Deck():
+ setArrayOfCards(c:Card):void
+getArrayOfCards(): const Card
+ PrintCard(c:Card):void
+ PrintValue(c:Card):void
+ PrintSuit(c:Card):void
+ shuffle():void
+ PrintAll():void
+ draw():Card

Player

- name:string
- total:int
- credit:float

+ Player():
+ Player(n:string):
+ setName(n:string): void
+ setTotal(*l:Hand , x : int):void
+ setCredit(c:float): void
+ getName():string
+ getTotal():int
+ getCredit():float
+ Player (&p:Player, c:float): friend operator+=

Card

+ name:CardName
+ suit:Suits
+ value:int

```
Start
  │
  ▼
srand(time(0))
  │
  ▼
Declared Variables:
Card cards, Deck deck, Player *p,
fstream rules, char *filename, string readLine,
int numOfP, int i, int j, int a=b=c=d, int index
  │
  ▼
Initialized Variables:
*fileName="Rules.txt",
a=b=c=d=2, index=51
  │
  ▼
rules.open(fileName, ios::in)
  │
  ▼
getline(rules, readLine)  ──True──►  cout << readLine <<endl;
  │
  ▼
restart="y";
  │
  ▼
answer!="N" && answer!="n"
  │
  ▼
restart=="y" || restart=="Y"
  │                    │                    │
True                  False                True
  ▼                    ▼                    ▼
cou<<"Exiting          2                restart="";
Programm... ";                              │
  │                                         ▼
  ▼                              cout << "How many players are
End                              playing? Only 3 can play at the "
                                 << "same time plus the dealer."
                                 << endl;
                                            │
                                            ▼
                                            1
```

```
                                                                    ( 1 )

                                                          True
cout <<"There's no one to                         numOfP<1 || numOfP>3
play with! Try again." << endl;      numOfP==0
cin>>numOfP;                                               False

                                                          p = new Player[4];

cout<<"Cannot enter              True
negative numbers, try        numOfP<0              cout << "Enter name
again."<<endl;                                     for the players." <<
cin>>numOfP;                                        endl;

cout <<"That's too many          True
people! Try again." << endl;  numOfP>3             i=1;i<=numOfP;i++      True     cin >> p[i].name
cin>>numOfP;
                                                          False

                                                   PlaceBets()
                                                   fuction is called     ( 2 )
                                                   to place bets

                                                   Set the name for the
                                                   dealer

                                                   Give credits to the
                                                   players

            False
                                                          True
load[i].setHand(deck,j)      True    j=0;j<2;j++           i=0;i<=numOfP;i++

                                                          False

                                                   cout<<endl;

                                                          ( 3 )
```

```
deck.PrintCard(load[1].getHand(j));
```

True

```
j=0;j<b;j++
```

False

```
cout<<p[1].getName() << "'s hand" << endl;
cout<<p[1].getName() << "'s Coin : "
<<p[1].getCredit()<< endl;
```

```
cout<<endl;
```

3

True

```
numOfP>=1
```

False

```
deck.PrintCard(load[2].getHand(j));
```

True

```
j=0;j<b;j++
```

False

```
cout<<p[2].getName() << "'s hand" << endl;
cout<<p[2].getName() << "'s Coin : "
<<p[2].getCredit()<< endl;
```

True

```
numOfP>=2
```

False

```
cout<<endl;
```

```
deck.PrintCard(load[3].getHand(j));
```

True

```
j=0;j<b;j++
```

False

```
cout<<p[3].getName() << "'s hand" << endl;
cout<<p[3].getName() << "'s Coin : "
<<p[3].getCredit()<< endl;
```

True

```
numOfP>=3
```

False

```
cout<<endl;
```

```
Dealer(deck,p,load,numOfP,a,b,c,d);
CheckSum(p,load,numOfP,b,c,d);
CheckWin(p,put,numOfP);
```

```
cout<<"Round "<<round<<"Ended. Play
another?"<<endl;
cout<<"'Y' for yes 'N' for no."<<endl;
cin>>answer;
round++;
```

4

```
4
```

answer!="N" && answer!="n" && answer!="Y" && answer!="y"

True

```
cout<<"Wrong input detected, only 'Y'
for yes, or 'N' for no is"
       <<"accepted. Try again."
       <<endl;
        cin>>answer;
```

False

```
cout<<"Would you like to reset game or
continue?"<<endl;
cout<<"'Y' for yes 'N' for no."<<endl;
cin>>restart;
```

False

restart=="n" || restart=="N"

False

restart=="y" || restart=="Y"

True

True

```
cout<<"OK! Round "
<< round << " starts
now!"<<endl<<endl;
```

```
delete []p;
```

```
deck.~Deck();
```

```
void Dealer(Deck &deck, Player *p, Hand *l, int numOfP, int &index,int &a, int &b, int &c,
int &d)
```

int choice,i,j;

numOfP>=1 —True→ cout << "Press 1 to let player one draw."<

numOfP>=2 —True→ cout << "Press 2 to let player two draw."<

numOfP>=3 —True→ cout << "Press 3 to let player three draw."<

False → cout <<"Press 4 to continue."<> choice;

choice!=4

(2)

False → (3)

choice==1 —False→ choice==2 —False→ choice==3

True ↓
p[1].hand[b] = draw(deck, index);
b++;

True ↓
p[2].hand[c] = draw(deck, index);
c++;

True ↓
p[3].hand[d] = draw(deck, index);
d++;

(1)

```
                              (1)                                                   (2)

          True                          true
numOfP>=1  ─────►  j=0;j<b;j++  ─────►  PrintCard(p[1].hand[j]);
                              False

          True                          true
numOfP>=2  ─────►  j=0;j<b;j++  ─────►  PrintCard(p[1].hand[j]);
                              False

          True                          true
numOfP>=3  ─────►  j=0;j<b;j++  ─────►  PrintCard(p[1].hand[j]);
          False                 False

cout << "Do you want to
draw more cards?"<

          True
cout << "Press 1 to let      numOfP>=1
player one draw."<
                              False

          True
cout << "Press 2 to let      numOfP>=2
player two draw."<
                              False

          True                 False
cout << "Press 3 to let      numOfP>=3
player three draw."<
                              False

                    cout <<"Press 4 to
                    continue."<> choice;


                                                                    False
(3)  ─►  int total=0;  ─►  cout<<endl<<endl<<p[0].name<<endl;  ─►  i=0;i<a;i++  ─►  p[0].total=total;
                                                                    True

                                                        PrintCard(p[0].hand[i]);              (4)
                                                        total+=p[0].hand[i].value;
```

```
( 4 )

p[0].total<17  --False-->  cout << p[0].name <<" has a value of " << total << endl;  --> End

p[0].total<17 --True-->
p[0].hand[a] = draw(deck, index);
PrintCard(p[0].hand[i])
total+=p[0].hand[a].value;
p[0].total=total;
a++;

void CheckSum(Player *p,int numOfP, int b, int c, int d)

int i;
int total=0;

numOfP>=1 --True--> p[1].total=0; --> i=0;i<b;i++ --True--> total+=p[1].hand[i].value;
i=0;i<b;i++ --False--> cout <<endl<< p[1].name <<" has a value of " << total <<endl<<endl;
--> p[1].total=total; total=0;

numOfP>=1 --False-->

numOfP>=2 --True--> p[2].total=0; --> i=0;i<c;i++ --True--> total+=p[2].hand[i].value;
i=0;i<c;i++ --False--> cout <<endl<< p[2].name <<" has a value of " << total <<endl<<endl;
--> p[2].total=total; total=0;

numOfP>=2 --False-->

numOfP>=3 --True--> p[3].total=0; --> i=0;i<d;i++ --True--> total+=p[3].hand[i].value;
i=0;i<d;i++ --False--> cout <<endl<< p[3].name <<" has a value of " << total <<endl<<endl;
--> p[3].total=total; total=0;

numOfP>=3 --False--> End

--> End
```

```
void CheckWin(Player *p,int numOfP)
```

False

End ← int i=1;i<=numOfP;i++

True

p[i].total<=21 —False→ cout<< p[i].name <<"'s hand busted! "<< " is out of the game."<<endl;

True

p[0].total<p[i].total || p[0].total>21 —True→ cout << p[i].name <<" wins!" << endl;

False

p[0].total==p[i].total —True→ cout << "Its a tie with the dealer with " << p[i].name <<"."<< " Play Again!"<<endl;

False

cout <<"The dealer wins against " <<p[i].name << "."<< endl;

void PrintCard(Card c)

```
void PrintCard(Card c)
    │
    ▼
┌──┤ PrintValue(c); ├──┐
    │
    ▼
  cout << " of ";
    │
    ▼
┌──┤ PrintSuit(c); ├──┐
    │
    ▼
  cout << endl;
    │
    ▼
   end
```

void PrintValue(Card c)

```
void PrintValue(Card c)
    │
    ▼
c.name==CardName::Ace ──True──> cout<<"Ace"; ──> c.value=11 ──┐
    │ False                                                    │
    ▼                                                          │
c.name==CardName::Jack ──True──> cout<<"Jack"; ──> c.value=10 ─┤
    │ False                                                    │
    ▼                                                          │
c.name==CardName::Queen ──True──> cout<<"queen"; ──> c.value=10┤
    │ False                                                    │
    ▼<─────────────────────────────────────────────────────────┘
  cout << c.value;
    │
    ▼
   End
```

void PrintSuit(Card c)

```
void PrintSuit(Card c)
    │
    ▼
c.suit==Suits::Clubs ──True──> cout<<"Ace"; ──┐
    │ False                                    │
    ▼                                          │
c.suit==Suits::Diamonds ──True──> cout<<"Ace";┤
    │ False                                    │
    ▼                                          │
c.suit==Suits::Hearts ──True──> cout<<"Ace"; ─┤
    │ False                                    │
    ▼                                          │
c.suit==Suits::Spades ──True──> cout<<"Ace"; ─┤
    │ False                                    │
    ▼<──────────────────────────────────────────┘
   End
```

void shuffle(Deck &d)

```
int n,m;
Card temp=d.arrCards[0];
```

```
n=0;n<52;n++
```

True →

```
m=rand()%52;
temp=d.arrCards[n];
d.arrCards[n]
d.arrCards[m]
d.arrCards[n]=d.arrCards[m];
d.arrCards[m]=temp;
```

False

End

void PrintAll(Deck d)

```
int col=(int)Suits::Clubs; col<=(int)Suits::Spades; col++
```

True →

```
int row=(int)CardName::Ace; row <= (int)CardName::King; row++
```

False

True

```
int index=
(13*col)+row-1;
```

```
PrintCard(d.arrCards[index]);
```

False

End

```
┌─────────────────────────────┐
│   void SetupCards(Deck &d)   │
└─────────────────────────────┘
              │
              ▼                                              False
      ╱─────────────────╲        True      ╱─────────────────────────────╲
     ╱ int col=(int)Suits::Clubs; col<= ╲──────────╲ int row=(int)CardName::Ace; row <= ╲
     ╲ (int)Suits::Spades; col++       ╱           ╲ (int)CardName::King; row++        ╱
      ╲─────────────────╱                           ╲─────────────────────────────╱
              │                                                   │
          False                                                True
              ▼                                                   ▼
   ┌─┬───────────────────┬─┐                      ┌────────────────────────────┐
   │ │    shuffle(d);    │ │                      │         Card c;             │
   └─┴───────────────────┴─┘                      │   c.suit = (Suits)col;      │
              │                                     │   c.name = (CardName)row    │
              ▼                                     └────────────────────────────┘
      ┌──────────────┐                                          │
      │     End      │                                          ▼               True
      └──────────────┘                               ╱───────────────────╲            ┌──────────────┐
                                                     ╱ c.name==CardName::Jack ╲────────│  c.value=10;  │
                                                     ╲───────────────────╱            └──────────────┘
                                                          │                                  │
                                                        flase         True                   │
                                                          ▼                                  │
                                                ╱───────────────────╲            ┌──────────────┐
   ┌──────────────────────────────┐           ╱ c.name==CardName::Queen ╲────────│  c.value=10;  │
   │  Card draw(Deck &d, int &index) │         ╲───────────────────╱            └──────────────┘
   └──────────────────────────────┘                │                                  │
              │                                   false         True                   │
              ▼                                     ▼                                   │
   ┌──────────────────────────────┐       ╱───────────────────╲            ┌──────────────┐
   │  return d.arrCards[index--];   │      ╱ c.name==CardName::Jack ╲────────│  c.value=10;  │
   └──────────────────────────────┘       ╲───────────────────╱            └──────────────┘
              │                                   │
              ▼                                 false
      ┌──────────────┐                            ▼
      │     End      │                  ┌──────────────────────────┐
      └──────────────┘                  │   c.value = (int)c.name;  │
                                        └──────────────────────────┘
                                                   │
                                                   ▼
                                        ┌──────────────────────────┐
                                        │  int index = (13*col)+row-1 │
                                        │  d.arrCards[index]=c        │
                                        └──────────────────────────┘
```