Uno

```
char *color;     //Color associated with Uno card
   int value;       //Value associated with Uno card
```

```
Uno(){color=new char [8];}      //Default constructor
   Uno(char *x,short y){color=x;   //Constructor 2 (overloading)
                  value=y;}
   Uno(const char []);                //Copy constructor
   virtual ~Uno(){delete [] color;}    //Destructor
   void operator=(const char[]);       //Assign card color to string
   void operator=(const short x);      //Assign value of card
   virtual bool operator==(const char x[]);    //Assign value of card
   virtual bool operator==(const short x);     //Assign value of card
   virtual char *getColor()const{return color;};//get color
   virtual int getValue(){return value;}  //returns value
   virtual void setColor(char *);         //Mutator-sets color of card
   virtual void setValue(int);            //Mutator-sets value of card
```

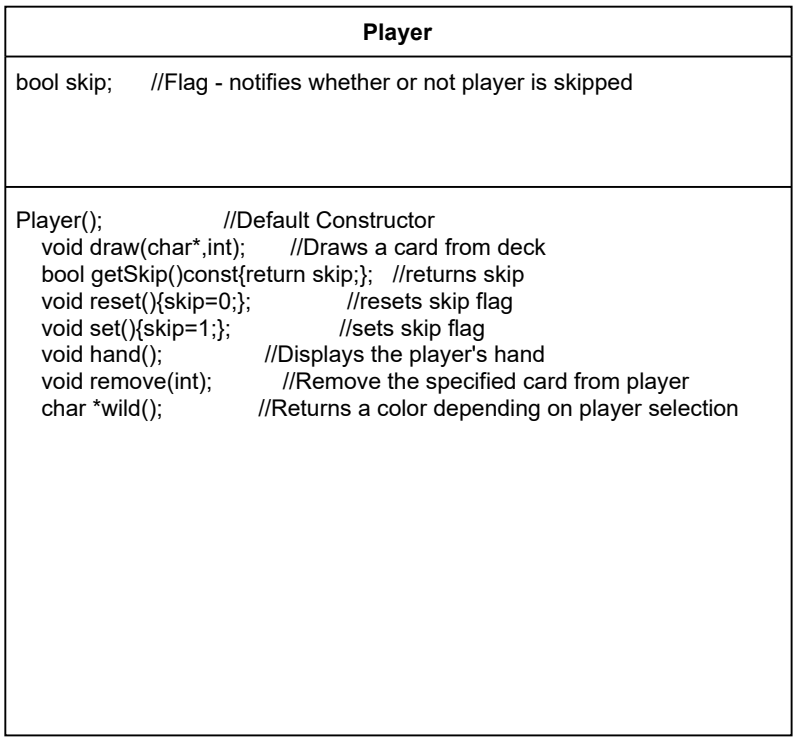
Pile

```
Uno *all;
short size;
```

```
Pile();                          //Pile Constructor
   virtual ~Pile(){delete [] all;};    //Pile Destructor
   virtual void modSize(short x){size+=x;};    //With a given value- size is
                                    //modified
   virtual void upSize(){size++;};     //Increments size
   virtual void downSize(){size--;};   //Decrements size
   virtual short getSize()const{return size;}; //Returns size
   virtual short getTotal()const{return 108;}; //Returns size of standard
                                    //Uno Deck
   virtual char *getColor(int i){return all[i].getColor();};//Returns color
   virtual short getValue(int i){return all[i].getValue();};//Returns value
   virtual void add(char*,int);    //Assigns a color an Uno card in Pile
   //Returns the card on top of the pile
   virtual void top(){cout<<getColor(size-1)<<" "<<getValue(size)<<endl;};
   //Overloaded operator, increments size
   virtual void operator++(){size++;}; //Increments size
```

Deck
bool stat[108]; //Status of Uno cards in deck

Deck(); //Default constructor
void define(); //Defines entire deck of Uno cards
void display(); //Displays entire deck of Uno cards
bool status(int); //Returns true if card is available, false otherwise
char *getColor(int)const;//Returns indicated card's color
short getValue(int)const;//Returns value of a card
void falsify(int); //Falsifies indicated index in bool array
int getIndex(); //Returns a valid index of a card that can be drawn
void reset(); //resets the stat array


Player
bool skip; //Flag - notifies whether or not player is skipped
Player(); //Default Constructor
void draw(char*,int); //Draws a card from deck
bool getSkip()const{return skip;}; //returns skip
void reset(){skip=0;}; //resets skip flag
void set(){skip=1;}; //sets skip flag
void hand(); //Displays the player's hand
void remove(int); //Remove the specified card from player
char *wild(); //Returns a color depending on player selection