```
template<class Sq>
class Game //classe abstraite
```

```
public:
Game(int,int); //dimensions
virtual void play();
virtual void demo();
virtual ~Game();

protected:
const int height;
const int width;
vector<Sq>* plateau;
long long score;
```

```
private:
bool quit;
virtual void init()=0;
virtual bool is_over() const=0;
virtual void move(Direction)=0;
virtual void print_board(ostream& o=cout) const;
friend ostream& operator<<(ostream& o, const Game<Sq>& game);
virtual void move_up();
virtual void move_down();
virtual void move_left();
virtual void move_right();
virtual bool is_stuck() const;
```

```
enum class Direction { up, down, left, right}
```

class Game_2048 : public Game<Square_2048>	template <class C> class Taquin : public Game<Square_Taquin<C>>	class Sokoban : public Game<CaseSok>
public: **Game_2048**(int height); protected: virtual Square_2048 **random_square**() const; virtual unsigned long long **random_value**() const; private: bool board_change; vector<pair<int, int>> empty_squares; virtual void **init**(); virtual void **move**(Direction dir); virtual bool **is_over**() const; void **transpose_board**(); void **pop_up_new_square**(); void **slide_line**(int i, Direction dir); void **merge_line**(int i, Direction dir); void **add_empty_square**(int i, int j); template<class It> int **slide_line_template**(It begin, It end); void **slide_board**(Direction dir, bool transpose); template<class It> void **merge_line_template**(It begin, It end); virtual bool **is_mergeable**(Square_2048& sq) const; virtual Square_2048 **merge**(Square_2048& sq);	public: static const Square_Taquin<C> empty; **Taquin**(int,int); virtual **~Taquin**(); private: int pos_empty_w; int pos_empty_h; virtual void **init**(); virtual bool **is_over**() const; virtual void **move**(); void **fill**(); void **mix**();	public: **Sokoban**(int h,int w, int nb_crates=-1); virtual **~Sokoban**(); protected: static const int min_height=10; static const int min_width=10; int nb_crates; int pos_h; int pos_w; int i_top_left; int j_top_left; int i_top_right; int j_top_right; int i_bottom_left; int j_bottom_left; int i_bottom_right; int j_bottom_right; virtual void **print_board**(ostream& o=cout) const; virtual void **init**(); virtual void **set_walls**(); virtual void **setExternalWalls**(); virtual void **setInternalWalls**(); virtual void **set_target_crates**(); virtual bool **free_zone**(int h_c, int l_c) const; virtual bool **outsideOfWalls**(int h_c, int l_c) const; virtual void **move**(Direction s); virtual void **set_pers**(); virtual bool **is_over**() const; virtual bool **is_stuck**() const;

class Game_2048_Num : public virtual Game_2048	class Game_2048_Neg : public virtual Game_2048
public: **Game_2048_Num**(int height, int base=2); protected: const int base; virtual unsigned long long **random_value**() const;	public: **Game_2048_Neg**(int height); protected: virtual Square_2048 **random_square**() const;

class Game_2048_Mult : public virtual Game_2048	class Game_2048_Dest : public virtual Game_2048
public: **Game_2048_Mult**(int height); protected: virtual Square_2048 **random_square**() const;	public: **Game_2048_Dest**(int height); protected: virtual Square_2048 **random_square**() const;

class Game_2048_Num2 : public virtual Game_2048	class Game_2048_Mix : public Game_2048_Num2, public Game_2048_Neg, public Game_2048_Mult, public Game_2048_Dest
public: **Game_2048_Num2**(int height); protected: const int base; virtual unsigned long long **random_value**() const;	public: **Game_2048_Mix**(int height, int base); protected: virtual Square_2048 **random_square**() const;

class Printable //classe abstraite
public: friend ostream& **operator**<<(ostream& out, const Printable& object); private: virtual void **print**(ostream& out) const = 0 ;

class Square_2048 : public Printable	template<class C> class Square_Taquin : public Printable

public:
static Square_2048 empty;
Square_2048(Square_2048_action action = empty, unsigned long long value =0);
bool **operator**==(const Square_2048& sq) const;
bool **operator!**=(const Square_2048& sq) const;
bool **dest_possible**(const Square_2048& sq) const;
bool **mult_possible**(const Square_2048& sq) const;
bool **is_opposite**(const Square_2048& sq) const;
bool **same_action**(const Square_2048& sq) const;
bool **same_value**(const Square_2048& sq) const;
Square_2048& **operator**=(const Square_2048& sq) const;
void **set_value**(unsigned long long value);
unsigned long long **get_value**() const;
void **swap**(Square_2048& sq);
bool **is_empty**() const;
 	
private:
Square_2048_action action;
unsigned long long value;
virtual void **print**(ostream& out) const;
	public:
	static const Square_Taquin<C> empty;
	Square_Taquin(unsigned long l=0);
	Square_Taquin(const Square_Taquin<C>& sq);
	bool **operator**==(const Square_Taquin<C>& sq) const;
	bool **operator!**=(const Square_Taquin<C>& sq) const;
	bool **operator**<(const Square_Taquin<C>& sq) const;
	bool **operator**<=(const Square_Taquin<C>& sq) const;
	bool **operator**>(const Square_Taquin<C>& sq) const;
	bool **operator**>=(const Square_Taquin<C>& sq) const;
	Square_Taquin& **operator**=(Square_Taquin<C>& sq);
	Square_Taquin& **operator++**();
	Square_Taquin& **operator++**(int);
	Square_Taquin& **operator--**();
	Square_Taquin& **operator--**(int);
	
	private:
	virtual void **print**(ostream& o) const;
	unsigned long value;

```
enum class Action_2048 { empty, none, neg, mult, div, destroy }
string to_string(Action_2048 action);
```

```
enum class CaseSok { empty, wall, pers, crate, target, crate_target, pers_target }
ostream& operator<<(ostream& out, CaseSok const& c);
```