```
class Jeu //classe abstraite
protected:
float** plateau;
const int longueur;
const int largeur;
static float T caseVide=0.0;
int score;

public:
Jeu(int,int); //dimensions
virtual void initialiser()=0;
virtual bool jeuTermine() const=0;
virtual void jouerHumain()=0;
virtual void jouerRobot()=0;
virtual void afficher(ostream& o=cout) const=0;
virtual void deplacerHaut()=0;
virtual void deplacerBas()=0;
virtual void deplacerGauche()=0;
virtual void deplacerDroite()=0;
virtual ~Jeu();
```

class Jeu2048 : public Jeu //les nombres seront entiers, mais pour pouvoir implémenter les variantes supplémentaires, on aura besoin de définir des constantes (qui ne correspondent pas à des nombres possibles) pour pouvoir afficher le plateau.	class Taquin : public Jeu	class Sokoban : public Jeu

private:
bool autresNombres;
bool nombresNegatifs;
bool multDiv;
bool destroy;
static const float MULT=1.0;
static const float DIV=-1.0;
static const float DESTROY_F=0,5;
//les valeurs 1, -1 et 0.5 n'étant pas des valeurs possibles de nombres, on les utilises pour représenter respectivement les cases x2, /2 et destroy)
 	
public:
Jeu2048(int l, int h, bool autres=false, bool neg=false, bool mult_div=false, bool destroy=false);
//par défaut, on implémente le jeu basique
~Jeu2048();
	private:
	int pos_x;
	int pos_y;
	//pos_x et pos_y représentent les coordonnées de la case vide
	
	public:
	Taquin(int,int);
	~Taquin();
		private:
		int pos_x;
		int pos_y;
		//pos_x et pos_y représentent les coordonnées de la case occupée par le personnage
		
		static const float PERS=1.0;
		static const float CAISSE=2.0;
		static const float BUT=3.0;
		static const float MUR=4.0;
		
		public:
		Sokoban(int,int);
		~Sokoban();

```
enum class CaseSok={pers,caisse,but,mur,vide}
```