Description des classes et variables/tableaux de classe :

```
Main {
    partie partie1 ;
}
Class partie {
    IGEchecs ige ;
    piece[8][8] plateau ;
    boolean arret ;
}
Class piece {
    protected char type ;
    protected boolean couleur;
}
Class pion extends piece {
    private static depl int[][] ;
}
Class tour extends piece {
    private static depl int[][] ;
}
Class cavalier extends piece {
    private static depl int[][] ;
}
Class fou extends piece {
    private static depl int[][] ;
}
Class dame extends piece {
    private static depl int[][] ;
}
Class roi extends piece {
    private static depl int[][] ;
}

Class erreurInstruction extends Exception {
}
```

Détails et justifications :

Note préalable : le programme ne permet pas de vérifier toutes les règles du jeu d'échec. La description des classes correspond à un premier jet et sera susceptible d'être modifiée dans la version finale du projet, pour ajouter de nouvelles fonctionnalités par exemple (vérification échec, mat, pat, roque, etc..)

La classe main initialise l'interface graphique puis ouvre une instance de partie. Ensuite, elle appelle partie.lancer

La classe partie :
- initialise ige, plateau[][] (mémorise la position des pièces sur le plateau) et arret
- Constructeur génère les pièces et les place au bon endroit dans plateau. Les cases vides de plateau sont initialisées à null.
- Méthode generePiece : Crée les pièces (initialisation, pion à dame) et les ajoute sur l'interface graphique
- lancer : Lance la boucle qui permet de jouer les tours successivement, arrêt lorsque variable arret passe à true.
- Méthode tour : Reçoit l'instruction d'un tour. Appelle verifSyntaxe et génère une erreurInstruction si erreur détectée. Si déclaration match nul, envoi à finPartie, sinon appel de analyserInstruction pour les deux joueurs dans un try et remet le plateau à son état de début de tour dans catch.
- Méthode verifSyntaxe : vérifie la structure de l'instruction avec regex et renvoie un booléen.
- Méthode analyserInstruction : Vérifie si instruction de roque ou autre instruction en notation complète ou abrégée. Si notation abrégée, appelle trouverPieceAbreg, sinon appelle piece.verifDeplacement. En cas de mat, appelle finPartie
- Méthode Roque : vérifie et effectue le roque, si erreur détectée soulève une erreurInstruction.
- Méthode pionADame : vérifie position du pion et transforme en dame (suppression du pion et appel de generePiece si condition respectée, sinon soulève une erreurInstruction.
- Méthode trouverPieceAbreg : recherche pièce correspondant à l'instruction dans plateau. Si aucune correspondance trouvée soulève une erreurInstruction, renvoi un int[] avec la position de la pièce dans plateau (colonne en 0, ligne en 1).
- Méthode finPartie : variable arret à true et affichage du texte de fin de partie.

La classe piece :
- private char type : type de pièce pour comparaison avec instruction.
- private boolean couleur
- constructeur reçoit et initialise les variables
- Méthode finalize : retire la pièce du plateau (variable plateau et ige) et supprime l'objet de la mémoire
- Méthode getType : renvoi le type
- Méthode getCouleur : renvoi la couleur
- Méthode verifDeplacement : vérifie déplacement en fonction du type de pièce et du coup, renvoi erreurInstruction en cas d'erreur, sinon appel de deplacer
- deplacer : déplace la piece, supprime pièce « attaquée » le cas échéant et met à jour plateau

Les classes pion, tour, fou, cavalier, dame, roi :
- depl[][] est un tableau bi-dimensionnel d'int qui référence : la portée de la pièce en 0 et les déplacements autorisés ensuite. Ex pour la dame : En 0 : {8} - En 1 : {1,1} pour caractériser le déplacement diagonal - En 2 : {0,1} pour le déplacement latéral. Note : Pour les pions, on se contente d'une valeur par défaut, leur déplacement sera déterminé dans la méthode deplacer.
- le constructeur appel super pour définir le type et la couleur
- une méthode getDepl renvoi depl[][]