1. Main
   * 게임을 시작하는 Main 클래스
   * main 메소드에서 MainFrame 객체 생성
2. MainFrame (JFrame)
   * 게임 시작 화면을 담당하는 MainFrame 클래스
   * BackgroundPanel, Menu, HowToPlay 객체 생성
3. BackgroundPanel (JPanel)
   * 게임 시작 화면의 panel을 담당하는 BackgroundPanel 클래스
4. Menu
   * Menu 클래스
   * Menu 객체 생성 시 2개의 버튼이 추가됩니다.
   * StartButton : JButton
     + 버튼 클릭 시
       - MainFrame을 종료합니다.
       - MapPlay를 생성합니다.
   * HtpButton : JButton
     + 버튼 클릭 시
       - backgroundPanel과 menu를 보이지 않게 합니다.
       - howToPlay를 보이게 합니다.
5. HowToPaly (JPanel)
   * 게임 방법을 띄워주는 HowToPaly 클래스입니다.
   * ExitButton : JButton
     + 버튼 클릭 시
       - HowToPlay를 보이지 않게 합니다.
       - backgroundPanel과 menu을 보이게 합니다.
6. MapPlay (JFrame)
   * 게임 frame인 MapPlay 클래스입니다.
   * MainFrame에서 startButton을 누르면 실행됩니다.
   * 게임 내의 객체들을 생성하고 키보드 입력을 받습니다.
   * MContext : MapPlay
     + MContext로 모든 객체에 접근이 가능합니다.
     + MapPlay내에서 생성되는 거의 모든 객체의 클래스가 같은 필드를 가지고 있습니다.
   * BackgroundMap : JLabel
     + 게임의 맵 이미지를 담당합니다.
   * Player 1, player 2 : Player
     + Player1과 player2는 각각 키보드의 a, w, d, s, shift 그리고 →, ←, ↑, ↓, space 키로 이동 및 공격이 가능합니다.
   * IsSpace, isShift : boolean
     + Player가 공격키를 꾹 누르고 있을 때 별사탕(총알)이 연발되지 않도록 해주는 필드입니다.
   * CoinDisplay1, coinDisplay2 : CoinDisplay
     + 각각 player1, player2의 coin을 보여주는 라벨입니다.
   * GameTimer : GameTimer
     + 게임에 타이머를 띄워주는 라벨입니다.
   * EnemyList : ArrayList<Enemy>
     + Enemy 리스트입니다.
   * EnemySize : int
     + EnemyList의 size 입니다.
   * State : int
     + 게임이 끝났는지, 게임중인지 상태를 나타내는 필드입니다.
     + 0 : 게임중, 1 : 게임 끝남
     + 게임 상태에 따른 입력 키의 제한을 줄 때 사용합니다.
   * Ending : Ending
     + 타이머가 끝나면 생성되는 ending 입니다.
   * BackgroundEndingImage : BackgroundEndingImage
     + 타이머가 끝나면 생성되는 backgroundEndingImage 입니다.
   * MapPlay()
   * InitObject() : void
   * InttSetting() : void
   * InitListener() : void
     + 키보드 리스너입니다.
     + State == 1인 경우 enter를 입력받으면 MapPly를 종료하고 MainFrame을 싱행해 시작 화면으로 돌아갑니다.
   * TimeOver() : void
     + Timer가 종료하면 호출되는 TimerOver 메소드입니다.
     + Ending과 backgroundEndingImage 객체를 생성합니다.
7. Moveable (interface)
   * 상하좌우 움직임과 공격 매소드를 인터페이스로 구현합니다.
   * Player, Enemy, Star에서 사용합니다.
8. Player (JLabel)
   * Player jlabel 클래스 입니다.
   * MContext : MapPlay
     + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
   * Type : int
     + Type, 1 : player 1, 2 : player 2
   * X, y : int
     + 위치 상태
   * PlayerWay : PlayerWay
     + 플레이어의 방향
   * Left, right, up, down : boolean
     + 움직임 상태
   * State : int
     + State, 0 : live, 1 : die
   * LeftWallCrash, rightWallCrash : boolean
   * SPEED : final int
     + Left, right 속력
   * JUMPSPEED : final int
     + Up, down 속력
   * HpPlayer : int
     + 플레이어의 채력
   * HpBar : HpBar
     + Player의 체력바, 체력바는 MapPlay가 아닌 Player 클래스 내에서 생성됨
   * CoinCnt
     + 플레이어의 코인 개수
   * Player1R, player1L, player1RD, player1LD, player2R, player2L, player2RD, player2LD : ImageIcon
     + Player1, player2의 각각 오른쪽, 왼쪽, 오른쪽 사망, 왼쪽 사망 imageicon
   * Timegap, pretime : long
     + Enemy와 추돌 시 1초간 enemy와 충돌해도 체력이 깎이지 않도록 하는 필드
   * Player(MapPlay mContext, int type)
     + 생성자
   * InitObject() : void
   * InitSetting(int type) : void
   * IntitBackfroundPlayerService() : void
     + BackgroundPlayerService 스레드를 열어 맵과의 충돌을 확인합니다.
   * Left(), right() : void
     + 키 입력이 들어오면 왼쪽, 오른쪽으로 움직입니다.
   * Up(), down() : void
     + 점프키가 들어오면 점프하고, 바닥이 없으면 하강합니다.
   * Attack() : void
     + 공격키를 누르면 실행됩니다.
     + Star 객체를 생성합니다.
   * Die() : void
     + 플레이어 사망 메소드입니다.
     + 이미지 변환 후 2초간 sleep 합니다.
     + DropAll() 메소드를 호출해 가지고 있는 코인의 반을 떨어뜨립니다.
     + MapPlay에서 삭제 후 revive()를 호출합니다.
   * Revive() : void
     + 플레이어 부활 메소드입니다.
     + 위치를 재설정한 후 MapPlay에 player를 다시 생성합니다.
     + 체력을 채우고 hpBar를 refill합니다.
   * Attacked() : void
     + 공격 받으면 실행되는 메소드입니다.
     + 체력을 1 감소시키고, 체력바를 수정합니다.
     + 체력이 0이라면, die() 메소드를 실행합니다.
   * Drop() : void
     + 코인을 하나 떨어뜨리는 메소드입니다.
     + 스레드를 열어 코인 객체를 생성합니다.
   * DropAll() : void
     + 가지고 있는 코인의 반만큼 drop() 메소드를 호출하고, 코인을 감소시킵니다.
9. PlayerWay (enum)
   * 플레이어의 방향 정보
   * LEFT, RIGHT
10. BackgroundPlayerService (implements Runnable)
    * 플레이어와 맵의 충돌을 확인해서 범위 밖으로 나가지 못하게 하고, 바닥을 인식하게 해주는 클래스입니다.
    * BufferdImage : image
      + 맵의 background 이미지를 저장합니다.
      + 색깔을 통해 외벽과 바닥을 확인하는 데에 쓰입니다.
    * Player : player
      + 해당 player의 위치를 통해 맵을 인식하고 player를 밖으로 나가지 못하게 합니다.
    * BackgroundPlayerService(Player player)
      + 생성자
    * Run() : void
      + While 문을 통해 바닥, 외벽과의 충돌을 확인합니다.
11. Enemy (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * X, y : int
      + 위치 상태
    * Left, right, up, down : boolean
      + 움직임 상태
    * State : int
      + State, 0 : live, 1 : die
    * Random : Random;
      + Move() 함수에서 오른쪽, 왼쪽 방향을 정하는 데에 사용되는 필드입니다.
      + Left(), right() 메소드에서 무작위로 점프하는 데에 사용되는 필드입니다.
    * EnemyWay : EnemyWay
      + Enemy의 방향
    * SPEED : final int
      + Left, right 속력
    * JUMPSPEED : final int
      + Up, down 속력
    * EnemyR, enemyL, enemyDie : ImageIcon
      + Enemy 오른쪽, 왼쪽, 사망 ImageIcon입니다.
      + EnemyR과 enemyL은 같은 이미지를 설정했지만, 구현은 그대로 놔뒀습니다.
    * enemy(MapPlay mContext)
      + 생성자
    * InitObject() : void
    * InitSetting(int type) : void
    * IntitBackfroundEnemyService() : void
      + IntitBackfroundEnemyService 스레드를 열어 맵과의 충돌을 확인합니다.
    * Move() : void
      + 처음 리스폰 시 오른쪽으로 갈 지 왼쪽으로 갈 지 결정하는 함수입니다.
    * Left(), right() : void
      + 왼쪽 벽 혹은 오른쪽 벽을 만나기 전까지 왼쪽 혹은 오른쪽으로 이동합니다.
      + 이동하며 player와 충돌이 있는지 확인하고 충돌하면 player를 공격합니다.
      + 랜덤으로 up() 함수를 호출합니다.
    * Up(), down() : void
      + 점프, 다운 메소드입니다.
    * Die() : void
      + Enemy가 star에 닿으면 호출됩니다.
      + 이미지를 사망 이미지로 바꾸고 drop() 메소드를 호출해 코인을 떨어뜨립니다.
      + 3초간 sleep 후 MapPlay에서 객체를 지우고 remakeEnemy()를 호출합니다.
    * RemakeEnemy() : void
      + 위치 재설정 후 enemy를 다시 MapPlay에 추가합니다.
      + Move() 함수를 통해 초기 방향을 정합니다.
    * Drop() : void
      + 스레드를 통해 coin 객체를 생성합니다.
12. EnemyWay
    * enemy의 방향 정보
    * LEFT, RIGHT
13. BackgroundEnemyService (implements Runnable)
    * enemy와 맵의 충돌을 확인해서 범위 밖으로 나가지 못하게 하고, 바닥을 인식하게 해주는 클래스입니다.
    * BufferdImage : image
      + 맵의 background 이미지를 저장합니다.
      + 색깔을 통해 외벽과 바닥을 확인하는 데에 쓰입니다.
    * Enemy : Enemy
      + 해당 player의 위치를 통해 맵을 인식하고 player를 밖으로 나가지 못하게 합니다.
    * BackgroundEnemyService (Enemy enemy)
      + 생성자
    * Run() : void
      + While 문을 통해 바닥, 외벽과의 충돌을 확인합니다.
14. Coin (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * Player1, player2 : Player
      + Player1, player2 객체를 받아 충돌을 확인합니다.
    * X, y : int
      + 위치 상태
    * Left, right, up, down : boolean
      + 움직임 상태
    * State : int
      + State, 0 : live, 1 : die
    * Random : Random;
      + Move() 함수에서 오른쪽, 왼쪽 방향을 정하는 데에 사용되는 필드입니다.
      + Left(), right() 메소드에서 무작위로 점프하는 데에 사용되는 필드입니다.
    * EnemyWay : EnemyWay
      + Enemy의 방향
    * SPEED : final int
      + Left, right 속력
    * JUMPSPEED : final int
      + Up, down 속력
    * EnemyR, enemyL, enemyDie : ImageIcon
      + Enemy 오른쪽, 왼쪽, 사망 ImageIcon입니다.
      + EnemyR과 enemyL은 같은 이미지를 설정했지만, 구현은 그대로 놔뒀습니다.
    * enemy(MapPlay mContext)
      + 생성자
    * InitObject() : void
    * InitSetting(int type) : void
    * IntitBackfroundEnemyService() : void
      + IntitBackfroundEnemyService 스레드를 열어 맵과의 충돌을 확인합니다.
    * Move() : void
      + 처음 리스폰 시 오른쪽으로 갈 지 왼쪽으로 갈 지 결정하는 함수입니다.
    * Left(), right() : void
      + 왼쪽 벽 혹은 오른쪽 벽을 만나기 전까지 왼쪽 혹은 오른쪽으로 이동합니다.
      + 이동하며 player와 충돌이 있는지 확인하고 충돌하면 player를 공격합니다.
      + 랜덤으로 up() 함수를 호출합니다.
    * Up(), down() : void
      + 점프, 다운 메소드입니다.
    * Die() : void
      + Enemy가 star에 닿으면 호출됩니다.
      + 이미지를 사망 이미지로 바꾸고 drop() 메소드를 호출해 코인을 떨어뜨립니다.
      + 3초간 sleep 후 MapPlay에서 객체를 지우고 remakeEnemy()를 호출합니다.
    * RemakeEnemy() : void
      + 위치 재설정 후 enemy를 다시 MapPlay에 추가합니다.
      + Move() 함수를 통해 초기 방향을 정합니다.
    * Drop() : void
      + 스레드를 통해 coin 객체를 생성합니다.
15. BackgroundCoinService
    * Image : BufferedImage
      + 충돌을 파악하기 위한 이미지를 담을 변수입니다.
      + 색깔을 이용하여 충돌을 판단합니다.
    * coin : Coin
      + 충돌을 파악할 코인의 정보를 담는 변수입니다.
    * checkUpWall() : boolean
      + 위쪽 벽과의 충돌을 감지합니다. 충돌하면 true을 return 합니다.
    * CheckDownWall() : boolean
      + 아래쪽 벽과의 충돌을 감지합니다. 충돌하면 true을 return 합니다.
    * checkLeftWall() : boolean
      + 왼쪽 벽과의 충돌을 감지합니다. 충돌하면 true을 return 합니다.
    * checkRightWall() : boolean
      + 오른쪽 벽과의 충돌을 감지합니다. 충돌하면 true을 return 합니다.
16. Star (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * player, playerEnemy : Player
      + player : 총알을 발사하는 주체
      + playerEnemy : player의 상대방
    * backgroundStarService : BackgroundStarService
      + 총알의 충돌 감지를 위한 클래스 타입의 변수입니다.
    * x , y : int
      + 위치 상태
    * left, right : boolean
      + 움직임 상태
    * state : int
      + State, 0 : 적을 맞출 수 있는 상태, 1 : 적을 맞출 수 없는 상태
    * Speed : int
      + 움직임 속력 상태
    * bullet : ImageIcon
      + 총알의 이미지 관련 변수
    * Star(MapPlay mContext, int type) : public
      + 생성자
    * initObject() : void
      + 총알 이미지 생성
      + backgroundBulletService 생성
    * initSetting() : void
      + 움직임 상태 변수들 초기화
      + 위치상태 좌표 초기화
      + 이미지를 bullet으로 설정
      + 크기는 50px \* 50px 로
      + State = 0 으로 초기화
    * left(), right() : void
      + 새로운 쓰레드를 생성합니다.
      + 1ms delay을 주며, 1ms마다 speed만큼의 x좌표가 변화합니다
      + 이것을 200번 반복하므로, 총알은 200\*speed만큼 이동합니다 (최대)
      + 이동중에 벽과 부딪치면 더 이상 움직이지 않습니다. 그 후 삭제됩니다.
      + 이동중에 상대방과 부딪치면, 상대방을 공격하는 attack()이 실행된 후, 더 이상 움직이지 않습니다. 그 후 삭제됩니다.
      + 이동중에 enemy와 부딧치면, enemy을 공격하는 attack(Enemy)이 실행된 후, 더 이상 움직이지 않습니다. 그 후 삭제됩니다.
      + 반복문이 종료되었다면, 그다음엔 clearStar()가 실행되어 객체가 삭제됩니다.
    * up() : void
      + 사용하지는 않으나 Moveable을 구현상속하므로, 아무런 내용없이 정의만 했습니다.
    * attack() : void
      + 자신을 삭제하는 clearStar()가 실행됩니다.
      + playerEnemy의 attacked()가 실행됩니다.
    * attack(Enemy enemy) : void
      + 자신을 삭제하는 clearStar()가 실행됩니다.
      + Enemy의 die()가 실행됩니다.
    * clearStar() : void
      + 메모리에서 자신을 삭제합니다.
      + 삭제의 적용을 위해서 다시 그립니다 (repaint())
17. BackgroundStarService
    * Image : BufferedImage
      + 충돌을 파악하기 위한 이미지를 담을 변수입니다.
      + 색깔을 이용하여 충돌을 판단합니다.
    * bullet : Star
      + 충돌을 파악할 코인의 정보를 담는 변수입니다.
    * checkLeftWall() : boolean
      + 왼쪽 벽과의 충돌을 감지합니다. 충돌하면 true을 return 합니다.
    * checkRightWall() : boolean
      + 오른쪽 벽과의 충돌을 감지합니다. 충돌하면 true을 return 합니다.
18. HpBar (JLabel)
    * Frame : final int
      + 크기 관련 상수입니다.
    * hpMax : final int
      + 최대 체력 관련 상수입니다.
    * inMax : final int
      + 무적 체력 관련 상수입니다.
    * barSize : int
      + 현재 체력에 비례하는 체력바 크기입니다.
    * maxBarSize : int
      + 최대 체력에 비례하는 체력바 크기입니다.
    * HpBar(int type)
      + 생성자
    * fill(int hp) : void
      + player객체의 체력 감소시, 체력에 비례한 공간을 감소시킵니다.
    * refill() : void
      + 체력바를 다시 새걸로 바꾸는 메서드입니다.
      + player객체가 부활할 때 실행됩니다.
    * paintComponent(Graphics g) : void
      + 그래픽 관련 메서드입니다.
19. CoinDisplay (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * type : int
      + 무슨 player인지를 알려주는 변수
    * player : Player
      + player : 이 객체의 소지 코인을 보여줍니다.
    * coinPlayerImage : ImageIcon
      + 이미지 관련 변수입니다.
    * textLabel : JLabel
      + 숫자를 띄우기 위한 레이블 객체입니다.
    * CoinDisPlay()
      + 생성자
    * initObject() : void
      + 이미지변수를 초기화 합니다.
    * initSetting() : void
      + textLabel을 설정합니다.
    * reset() : void
      + player의 코인이 변화하면 player의 coin의 개수만큼 textLabel의 text을 설정해 줍니다.
20. GameTimer (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * seconds, minutes : int
      + 초, 분에 대한 변수.
    * backgroundTimer : ImageIcon
      + 이미지 관련 변수입니다.
    * textLabel : JLabel
      + 숫자를 띄우기 위한 레이블 객체입니다.
    * setGameTime : final int
      + 게임 플레이 시간에 대한 변수입니다.
    * GameTimer(MapPlay mContext)
      + 생성자
    * initObject() : void
      + 이미지변수를 초기화 합니다.
    * initSetting() : void
      + textLabel을 설정합니다.
    * updateTime() : void
      + 시간을 업데이트 합니다 (textLabel의 text을 줄어든 시간만큼으로 바꿔줍니다)
    * initSetting() : void
      + textLabel을 설정합니다.
    * TimerListener implements ActionListener : class
      + ActionListenenr을 구현상속하는 클래스입니다.
      + 시간초의 감소와, 게임의 종료를 담당합니다.
      + 1초마다 second을 감소시킵니다
      + second가 음수가 되면 minute을 감소하고 second = 59로 만듭니다.
      + 1초마다 updataTime()을 해줍니다.
      + second와 miutes가 둘다 0이면 게임을 종료하는 mContext.Timeover()을 실행합니다.
21. Ending (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * ending1, ending2, ending3, ending4, ending5, ending6, ending7, ending8, ending9, ending10, endingNull : ImageIcon
      + 엔딩화면을 점멸하게 만드는 변수입니다.
    * Type : int
      + 게임의 결과를 저장합니다.
    * Ending(MapPlay mContext,int type)
      + 생성자
    * initObject() : void
      + type 변수에 저장된 결과에 따라 엔딩화면을 저장하는 ending1~10까지의 ImageIcon 변수에 설정할 이미지를 저장합니다.
      + EndingNull에 이미지를 저장합니다.
    * initSetting() : void
    * -run() :void
      + 스레드를 사용해 0.075초 간격으로 이미지가 바뀌며 점멸하도록 합니다.
    * tryCatch() : void
22. BackgroundEndingImage (JLabel)
    * MContext : MapPlay
      + MContext로 MapPlay 내에서 생성된 모든 객체에 접근 가능
    * backgroundEnding : ImageIcon
      + 엔딩시 배경화면을 저장합니다.
    * BackgroundEndingImage(MapPlay mContext)
      + 생성자
    * initObject() : void
      + backgroundEnding에 이미지를 저장합니다.
    * initSetting() : void
      + 배경화면을 설정합니다.