**package** hw3;

**import** java.util.Random;

**import** api.Generator;

**import** api.Icon;

**import** api.Piece;

**import** api.Position;

/\*\*

\* Generator for Piece objects in BlockAddiction. Icons are always selected

\* uniformly at random, and the Piece types are generated with the following

\* probabilities:

\*

\* LPiece - 10%

\* DiagonalPiece - 25%

\* CornerPiece - 15%

\* SnakePiece - 10%

\* IPiece - 40%

\*

\* The initial position of each piece is based on its vertical size as well as

\* the width of the grid (given as an argument to getNext). The initial column

\* is always width/2 - 1. The initial row is: \*

\*

\* LPiece - row = -2

\* DiagonalPiece - row = -1

\* CornerPiece - row = -1

\* SnakePiece - row = -1

\* IPiece - row = -2

\*

\*/

**public** **class** BasicGenerator **implements** Generator {

**private** Random rand;

/\*\*

\* Constructs a BasicGenerator that will use the given Random object as its

\* source of randomness.

\*

\* **@param** givenRandom instance of Random to use

\*/

**public** BasicGenerator(Random givenRandom) {

rand = givenRandom;

}

@Override

**public** Piece getNext(**int** width) {

**int** n = rand.nextInt(101);

**int** col = width / 2 - 1;

**if** (n > 0 && n <= 10) {

**return** **new** LPiece(**new** Position(-2, col), icons);

} **else** **if** (n > 10 && n <= 35) {

**return** **new** DiagonalPiece(**new** Position(-1, col), icons);

} **else** **if** (n > 35 && n <= 50) {

**return** **new** CornerPiece(**new** Position(-1, col), icons);

} **else** **if** (n > 50 && n <= 60) {

**return** **new** SnakePiece(**new** Position(-1, col), icons);

} **else** **if** (n > 60 && n <= 100) {

**return** **new** IPiece(**new** Position(-2, col), icons);

} **else** {

**return** **null**;

}

}

@Override

**public** Icon randomIcon() {

**return** **new** Icon(Icon.COLORS[rand.nextInt(Icon.COLORS.length)]);

}

}