```
import java.util.*;

public class PasswordGenerator {

    private static final char[][] KEYBOARD = {
            {'1', '2', '3', '4', '5', '6', '7', '8', '9', '0'},
            {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'},
            {'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l'},
            {'z', 'x', 'c', 'v', 'b', 'n', 'm'}
    };

    private static Map<Character, List<Character>> validMovesMap = new HashMap<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the first character: ");
        char firstChar = scanner.next().charAt(0);

        if (isValidCharacter(firstChar)) {
            initializeValidMovesMap();
            String password = generatePassword(firstChar);
            System.out.println("Generated password: " + password);
        } else {
            System.out.println("Invalid character. Please enter a digit or a lowercase English letter.");
        }
    }
```

```
private static boolean isValidCharacter(char ch) {
    return (ch >= '0' && ch <= '9') || (ch >= 'a' && ch <= 'z');
}

private static String generatePassword(char startChar) {
    StringBuilder password = new StringBuilder(String.valueOf(startChar));
    Random random = new Random();

    for (int i = 1; i < 8; i++) {
        char lastChar = password.charAt(i - 1);
        List<Character> validMoves = validMovesMap.get(lastChar);

        if (validMoves != null && !validMoves.isEmpty()) {
            int randomIndex = random.nextInt(validMoves.size());
            password.append(validMoves.get(randomIndex));
        } else {
            password.append(startChar);
        }
    }

    return password.toString();
}

private static void initializeValidMovesMap() {
    for (int i = 0; i < KEYBOARD.length; i++) {
        for (int j = 0; j < KEYBOARD[i].length; j++) {
```

```
                char key = KEYBOARD[i][j];
                List<Character> validMoves = new ArrayList<>();
                populateValidMoves(i, j, key, validMoves);
                validMovesMap.put(key, validMoves);
            }
        }
    }

    private static void populateValidMoves(int x, int y, char currentKey, List<Character> validMoves) {
        for (int i = 0; i < KEYBOARD.length; i++) {
            for (int j = 0; j < KEYBOARD[i].length; j++) {
                char otherKey = KEYBOARD[i][j];
                int distance = calculateDistance(x, y, i, j);
                if (distance >= 2 && distance <= 3) {
                    validMoves.add(otherKey);
                }
            }
        }
    }

    private static int calculateDistance(int x1, int y1, int x2, int y2) {
        return Math.abs(x1 - x2) + Math.abs(y1 - y2);
    }
}
```