백준 1697 숨바꼭질

```java
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Baekjoon_1697 { // 숨바꼭질

    static int N;
    static int K;
    static int[] check;

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        N = sc.nextInt(); // 수빈이의 위치 입력
        K = sc.nextInt(); // 동생의 위치 입력
        check = new int[100001]; // 가장 빠른 경우를 구하는 문제이므로 방문체크 배열

        bfs(N); // bfs

        System.out.println(check[K] - 1); // 첫 시작지점이 1 이므로 도착지점은 -1을 해 주어야 한다.
    }

    private static void bfs(int number) {

        Queue<Integer> q = new LinkedList<Integer>();

        q.add(number); // 첫 시작지점을 큐에 넣고
        check[number] = 1; // 첫 시작지점을 방문체크 하기
```

```java
        while (!q.isEmpty()) { // 큐가 빌 때까지 반복하기

            int temp = q.poll(); // 큐에 값을 temp에 저장, temp는 이동위치라고 생각

            if (temp == K) // 이동위치가 동생위치랑 같으면 반복문 탈출
                break;

            if (temp - 1 >= 0 && (check[temp - 1] == 0)) { // 이동가능범위 인지 확인 후에 방문체크 확인 조건
                q.add(temp - 1); // 이동할 위치를 큐에 입력
                check[temp - 1] = check[temp] + 1; // 이동할 위치에 현재위치 +1을 해주어 방문체크 AND 걸린 시간까지 확인
            }

            if (temp + 1 <= 100000 && (check[temp + 1] == 0)) { // 이동가능범위 인지 확인 후에 방문체크 확인 조건
                q.add(temp + 1); // 이동할 위치를 큐에 입력
                check[temp + 1] = check[temp] + 1; // 이동할 위치에 현재위치 +1을 해주어 방문체크 AND 걸린 시간까지 확인
            }

            if (temp * 2 <= 100000 && (check[temp * 2] == 0)) { // 이동가능범위 인지 확인 후에 방문체크 확인 조건
                q.add(temp * 2); // 이동할 위치를 큐에 입력
                check[temp * 2] = check[temp] + 1; // 이동할 위치에 현재위치 +1을 해주어 방문체크 AND 걸린 시간까지 확인
            }
        }
    }
}
```