```
import java.util.ArrayList;
import java.util.Scanner;

public class BestFitMemoryAllocation {

  // Define MemoryBlock as a static nested class
  static class MemoryBlock {
    int size;
    boolean isAllocated;

    MemoryBlock(int size) {
      this.size = size;
      this.isAllocated = false;
    }
  }

  private ArrayList<MemoryBlock> memoryBlocks;

  // Constructor to initialize memory blocks
  public BestFitMemoryAllocation(ArrayList<Integer> blockSizes) {
    memoryBlocks = new ArrayList<>();
    for (int size : blockSizes) {
      memoryBlocks.add(new MemoryBlock(size));
    }

    // Sort memory blocks in ascending order to make the allocation easier
    memoryBlocks.sort((a, b) -> a.size - b.size);
  }
```

```
public boolean allocateMemory(int requestedSize) {
    int bestFitIndex = -1;
    int minWaste = Integer.MAX_VALUE;

    // Find the best fit block
    for (int i = 0; i < memoryBlocks.size(); i++) {
        MemoryBlock block = memoryBlocks.get(i);
        if (!block.isAllocated && block.size >= requestedSize) {
            int waste = block.size - requestedSize;
            if (waste < minWaste) {
                minWaste = waste;
                bestFitIndex = i;
            }
        }
    }

    if (bestFitIndex == -1) {
        System.out.println("No suitable block found for size " + requestedSize);
        return false;
    }

    // Allocate the best-fit block
    MemoryBlock bestFitBlock = memoryBlocks.get(bestFitIndex);
    bestFitBlock.isAllocated = true;
    System.out.println("Allocating " + requestedSize + " to block of size " + bestFitBlock.size);

    // Optionally split the block if there's leftover space
    if (bestFitBlock.size > requestedSize) {
```

```
        memoryBlocks.add(new MemoryBlock(bestFitBlock.size - requestedSize)); // Create a new free block
        System.out.println("Remaining free block of size " + (bestFitBlock.size - requestedSize));
      }
      return true;
    }

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

      // Take user input for memory block sizes
      System.out.print("Enter number of memory blocks: ");
      int n = sc.nextInt();
      ArrayList<Integer> blocks = new ArrayList<>();
      System.out.println("Enter the sizes of memory blocks:");
      for (int i = 0; i < n; i++) {
        blocks.add(sc.nextInt());
      }

      BestFitMemoryAllocation allocator = new BestFitMemoryAllocation(blocks);

      // Take user input for memory requests
      System.out.print("Enter number of memory requests: ");
      int m = sc.nextInt();
      System.out.println("Enter the sizes of memory requests:");
      for (int i = 0; i < m; i++) {
        int requestSize = sc.nextInt();
        allocator.allocateMemory(requestSize); // Try to allocate the memory
      }
```

```
        sc.close();
    }
}
```

```
java -cp /tmp/txzABboV2J/BestFitMemoryAllocation
Enter number of memory blocks: 3
Enter the sizes of memory blocks:
2
6
4
Enter number of memory requests: 3
Enter the sizes of memory requests:
5
Allocating 5 to block of size 6
Remaining free block of size 1
1
Allocating 1 to block of size 1
8
No suitable block found for size 8

=== Code Execution Successful ===
```