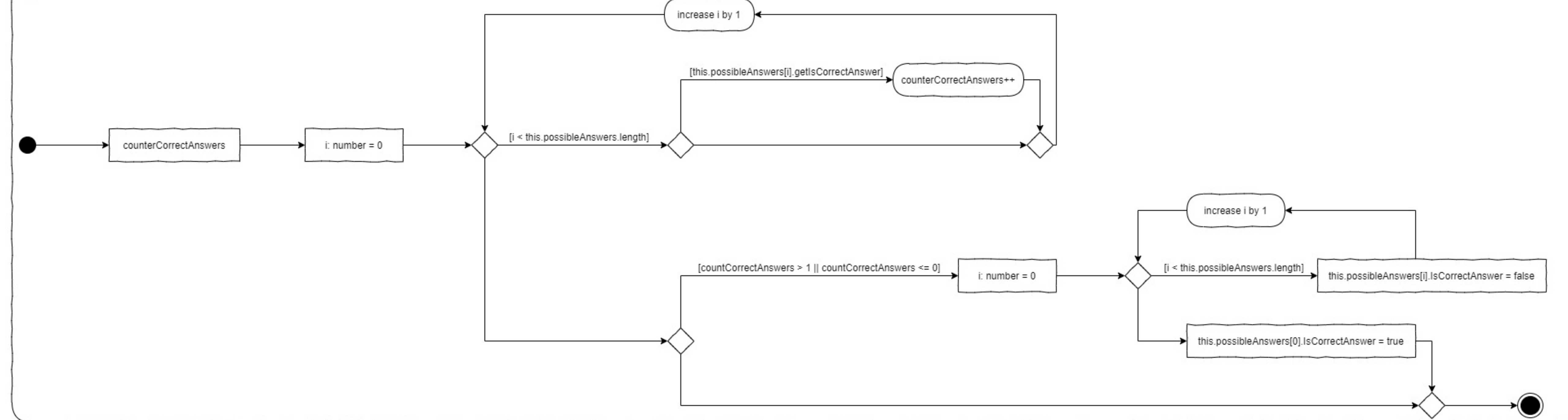
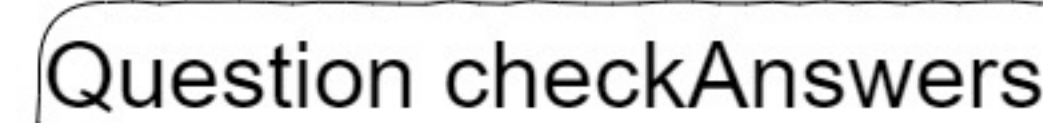


Input: Kimi no Na wa y

Output: Richtig

Output: Richtig beantwortete Fragen: 1



```

    graph LR
      Start(( )) --> Prompt1(prompt to question, default: "Welcher der folgenden Begriffe bezeichnet keine große Saurierepoche der Weltgeschichte?")
      Prompt1 --> NewQuestionText1(newQuestionText: string)
      NewQuestionText1 --> ParseInt1(parseInt(prompt to question, default))
      ParseInt1 --> AnswerCount1(answerCount: number)
      AnswerCount1 --> IsNaN1{isNaN(answerCount) || answerCount > 6 || answerCount <= 0}
      IsNaN1 --> AnswerCount2(answerCount = 2)
      AnswerCount2 --> IsNaN2{isNaN(answerCount) || answerCount > 6 || answerCount <= 0}
      IsNaN2 --> NewQuestion2(new Question(newQuestionText, new Array(answerCount)))
      NewQuestion2 --> I0(i: number = 0)
      I0 --> IsLess1{if < answerCount}
      IsLess1 --> Prompt2(prompt to question, default)
      Prompt2 --> NewAnswerText(newAnswerText: string)
      NewAnswerText --> IsLess2{if < answerCount}
      IsLess2 --> NewPossibleAnswers1(newQuestion.possibleAnswers[i] = new Answer(newAnswerText, true))
      IsLess2 --> NewPossibleAnswers2(newQuestion.possibleAnswers[i] = new Answer(newAnswerText, false))
      NewPossibleAnswers1 --> IncreaseI1(increase i by 1)
      NewPossibleAnswers2 --> IncreaseI1
      IncreaseI1 --> IsLess1
      IsLess1 --> CheckAllAnswers(newQuestion.checkAllAnswers)
      CheckAllAnswers --> Dispatch(dispatch)
      Dispatch --> Push(preDefinedQuestions.push(newQuestion))
      Push --> End(( ))
  
```

The diagram illustrates the logic for generating a new question. It starts with a prompt to the user, which is then parsed to determine the number of possible answers. If the number is not a valid integer or is outside the range of 1 to 6, it defaults to 2. A new question is then generated with a specified number of possible answers. The process then enters a loop where each possible answer is evaluated against the question. If an answer is correct, it is marked as such. The loop continues until all possible answers have been evaluated. Finally, the question is added to a predefined list of questions.

```

graph LR
    Start(( )) --> Init[randomQuestionInd: number = Math.floor(Math.random() * preDefinedQuestions.length)]
    Init --> GetQuestion[randomQuestion: Question = preDefinedQuestions[randomQuestionInd]]
    GetQuestion --> GetAnswersLength[preDefinedQuestions.splice(randomQuestionInd, 1)]
    GetAnswersLength --> SetAnswersLength[randomQuestionAnswersLength: number = randomQuestion.possibleAnswers.length]
    SetAnswersLength --> SetAnswersString[randomQuestionAnswersAsString: string = ]
    SetAnswersString --> Dispatch[dispatch]
    Dispatch --> SetIndex[i: number = 0]
    SetIndex --> LoopStart(( ))
    LoopStart --> IsEnd{[i = randomQuestionAnswersLength]}
    IsEnd --> IncreaseI[Increase i by 1]
    IncreaseI --> LoopStart
    IsEnd --> BuildAnswers[randomQuestionAnswersAsString += "Antwort: " + (i + 1) + " = " + randomQuestion.possibleAnswers[i].answerText + "<br>"]
    BuildAnswers --> Parse[parseInt(prompt to question, default)]
    Parse --> SetUserAnswer[userAnswer: number]
    SetUserAnswer --> IsValid{[isNaN(userAnswer) || userAnswer > randomQuestionAnswersLength || userAnswer <= 1]}
    IsValid --> SetUserAnswer1[userAnswer = 1]
    SetUserAnswer1 --> IsValid
    IsValid --> Output{ }
    Output --> OutputRichtig[output: "richtig"]
    Output --> OutputFalsch[output: "falsch"]
    OutputRichtig --> CorrectAnswers[correctAnswers++]
    OutputFalsch --> CorrectAnswers
    CorrectAnswers --> GetChoice[getUserChoice]
    GetChoice --> End(( ))

```