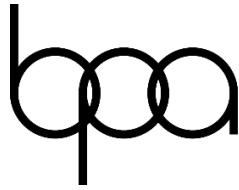


Contestant ID: \_\_\_\_\_

Time: \_\_\_\_\_

Rank: \_\_\_\_\_



**BUSINESS  
PROFESSIONALS  
of AMERICA**  
Giving Purpose to Potential

# **JAVA PROGRAMMING**

## **(340)**

## **REGIONAL 2026**

**APPLICATION KNOWLEDGE:**

TOTAL POINTS \_\_\_\_\_ (515 points)

**TEST TIME: 90 minutes**

### GENERAL GUIDELINES.

Failure to follow any of these rules may result in disqualification:

1. **Submission Requirements:** Contestants must submit this test booklet along with any printouts.
2. **Permitted Items:** Only the equipment, supplies, and materials specified for this event are allowed in the testing area. Previous BPA tests and sample tests (whether handwritten, photocopied, or typed) are not permitted.
3. **Electronic Devices:** Electronic devices will be monitored according to ACT standards.

**Your name and/or school name should not appear on work you submit for grading.**

1. Create a folder on the flash drive provided using your **Contestant ID** as the name of the folder.
2. Copy your entire solution/project into this folder so that the graders may open your project to review the source code and run your solution.

**\*Submissions that do not contain source code will not be graded.**

3. Ensure that the files required to run your program are present and **will execute** on the flash drive provided.

\*Advanced IDE's will place your files into multiple folders that are not centralized. This could make grading on third party machines impossible. The graders **will not compile or alter your source code** to correct for not including all solution files in the contestant number folder you created.

### **Commenting for Source Code Review:**

1. Certain sections of your code will be graded.
2. These gradable blocks of code can range from creating data structures, method algorithms, exception handling, and class construction.
3. Code Commenting Requirements:
  - a. Clear and concise comments must be included for all major components of the program, specifically:
  - b. Input: Document on how data is received or acquired, including user inputs, file reads, the format of the input, API requests, etc.
  - c. Processing: Provide comments that explain the logic, algorithms, or transformations applied to the input data.
  - d. Output: Describe how and where the final results are produced, the format of the output, whether through user interfaces, files, or other systems.
4. These comments should enhance understanding of the program's flow and intent, making it easier for others to read, maintain, and debug the code.
5. The grading rubric contains a section called Source Code Review: in this section are listed descriptions of all the graded programming concepts.
6. Each gradable item and any significant programming area should be commented on. There will be points awarded for proper commenting.

### Arithmetic Practice

You are going to create a program to help elementary students practice arithmetic. It will be saved as **ArithmeticPractice.java**.

Your **Contestant ID** must be included in a comment at the top of the program.

### Assumptions You Can Make

- Users will only input **valid** information when prompted for input. Remember that an incorrect answer to a problem is still valid input.
- The program will only exit when the user correctly answers the number of arithmetic problems the user asked for.

### Input

You must use the exact variable names listed below with the last **2 digits of your Contestant ID** added to the end of each where you see the **xx**.

**String userNamexx**

**int arithmeticTypexx, highestNumberxx, requestedCorrectNumberxx**

ArithmeticPractice will use a **Scanner** and the exact prompts shown below to get user input:

**What is your name: *Student 1***

**What would you like to practice?**

**1-addition, 2-subtraction, 3-multiplication or 4-division: *4***

**Highest number you would like included in the practice problems: *10***

**Number of problems needed to be answered correctly before your practice session ends:  
*5***

Once the initial input is collected, the message below is outputted based on **userNamexx** and **requestedCorrectNumberxx**.

**Student 1, please input the answer to each problem until you answer 5 correctly. Good luck!**

### Arithmetic Practice Problems Execution

After collecting all the input, you will use a **while** loop that runs until **int currentNumberCorrectxx** equals the number of problems needed to be answered correctly that the user inputted (**requestedNumberCorrectxx**):

During each iteration of the while/ do while loop:

- 2 random numbers (**int number1xx** & **int number2xx**) between **0** & **highestnumberxx** inclusive (0 and **highestnumberxx** should both be able to be randomly selected) will be selected
- 1 **switch** statement which will be used to:
  - Set **String operationsymbolxx**, which will be outputted as part of the problem the user will see based on **arithmetictypexx** the user inputted
    - \*for example if the user inputs 1:
 

**operationSymbolxx = “+”**
  - Set **int correctanswerxx** to the correct answer based on **arithmetictypexx**
    - \*for example if the user inputs 1:
 

**correctAnswerxx = number1xx + number2xx**
  - If the user selects 4 (division) you will need to find both the **quotient and the remainder (modulus (%))** (**int correctremainderxx**) since this is for elementary students
  - For case 4 you will need a while loop that keeps picking a random number for **number2xx** while it equals 0 because you can't divide by 0
- Once 2 random numbers have been selected and **if/else** will be used to output the problem as the prompt for the user to input the answer
  - In the case of division, the user needs to be prompted for both **int useranswerxx** & **int userremainderxx**
  - **Inside both the if and else**, an **if statement** must be used to check whether the user inputs the correct answer into **int useranswerxx**
- Random problems using the same arithmetic operation the user selected as part of the initial input and two newly generated random numbers will continue to be displayed for the user to answer until **numbercorrectxx equals requestedcorrectnumberxx**

### Final Output

- once the user has answered **requestedCorrectNumberxx** correctly, the program will **exit the loop based on the loop condition instead of by using a break statement**
- the following message containing the **user's name and the number of correct answers** will be outputted (this output is based on the sample input in the input section above)
- the Scanner must be closed

**Congratulations, Student 1, you answered 5 arithmetic problems correctly!**

### Sample Runs

**\*The dialogs must match the following exactly except that the numbers in the problems will be random:**

**What is your name:** *Student 2*

**What would you like to practice?**

**1-addition, 2-subtraction, 3-multiplication or 4-division: 2**

**Highest number you would like included in the practice problems: 5**

**Number of problems needed to be answered correctly before your session ends: 2**

**Student 2, please input the answer to each problem until you answer 2 correctly. Good luck!**

**5 – 0 = 5**

**Correct!**

**4 – 3 = 1**

**Incorrect. Try again!**

**2 - 3 = -1**

**Correct!**

**Congratulations, Student 2, you answered 2 arithmetic problems correctly!**

If the user picks 4 (division) a dialog like the following should take place (the first is answered correctly and the second isn't):

**What is your name: *Student 3***

**What would you like to practice?**

**1-addition, 2-subtraction, 3-multiplication or 4-division: 4**

**Highest number you would like included in the practice problems: 10**

**Number of problems needed to be answered correctly before your practice session ends:  
3**

**Student 3, please input the answer to each problem until you answer 3 correctly. Good luck!**

**Quotient of: 10 / 4 = 2**

**Remainder of: 10 / 4 = 1**

**Incorrect. Try again!**

**Quotient of: 5 / 8 = 0**

**Remainder of: 5 / 8 = 5**

**Correct!**

**Quotient of: 0 / 3 = 0**

**Remainder of: 0 / 3 = 3**

**Incorrect!**

**Quotient of: 7 / 6 = 1**

**Remainder of:  $7 / 6 = 1$**

**Correct!**

**Congratulations, Student 3, you answered 3 arithmetic problems correctly!**

### **Main Requirements**

1. The program must be named **ArithmeticPractice.java** and all solution files must be on the flash drive in a folder with your **Contestant ID** as the folder name.
2. Your **Contestant ID** must be included in a comment at the top of the program.
3. Your code must contain the **10 required comments** from the directions.
4. Your code must use the **exact variable names** given in the directions followed by the last 2 digits of your contestant number in place of **xx**.
5. Your code must use a **Scanner** to collect all input.
6. Your code must use the **exact prompts** given in the directions.
7. Your code must produce the **exact output** given in the directions.
8. Your code must contain **2 while loops** (one for generating the problems and one to make sure the program doesn't attempt to divide by 0).
9. Your code must use **modulus (%)** to find the remainder of any division problems.
10. Your code must contain a **switch statement** to assign the operator and the answer to each problem.
11. Your code must contain **2 if/else statements**:
  - 1 to **output the problem** based on whether the user chose **division** or **1 of the other 3 operations**
  - 1 to **output the correct message** based on if the inputted answer was correct
12. Your code must contain **2 if statements** to check whether the user answered the problem correctly based on whether it is **division** or **1 of the other 3 operations**.

<b>Solution and Project</b>		
The project is present on the flash drive		20 points
The program file is in one folder: the folder name is your Contestant ID		10 points

<b>Program Execution</b>		
Code copied to USB drive and the program runs from USB <i>If the program does not execute, then the remaining items in Program Execution receives a score of zero.</i>		20 points
Allows user to input name, arithmetic type, highest number allowed, and required number of correct answers successfully.		20 points
Prompts are clearly worded and match exactly the required prompts from the problem description above.		20 points
Correctly displays the user's name and number of required correct answers in the instruction message after collecting input.		20 points
Program generates arithmetic problems that align with selected arithmetic type. Random numbers are within the valid range (0 to highest number inclusive).		20 points
In division mode, program avoids division by zero by re-generating number2 when zero is selected.		10 points
Arithmetic is performed correctly for each operation type: +, -, *, and both quotient and remainder for ÷.		20 points
Accepts and correctly validates user answers, including both quotient and remainder for division.		20 points
Displays <b>"Correct!"</b> when the answer is right and <b>"Incorrect. Try again!"</b> when it is wrong. Feedback is immediate and repeats new problems for incorrect answers.		20 points
Only increments the number of correct answers when an answer is completely correct. Keeps asking new problems until the correct target is met.		20 points
Program ends only after user reaches the exact number of correct answers requested. Loop control must behave exactly as described.		10 points
Displays correct final message including user name and number of problems answered correctly, matching required format.		10 points



<b>Source Code Review</b>		
Program is properly commented (See Commenting for Source Code Review above)		50 points
Variable Names (check only one and award the corresponding points)		
<input type="checkbox"/> All variables are named exactly as in the directions & the <b>xx in each has been changed</b> to the last 2 digits of the contestant number. (30 points) <input type="checkbox"/> All variables are named exactly as in the directions but the <b>xx in each has not been changed</b> to the last 2 digits of the Contestant ID. (20 points) <input type="checkbox"/> Some variables are named as in the directions. (10 points) <input type="checkbox"/> None of the variables are named as in the directions. (0 points)		30 points
Code to <b>prompt</b> for the <b>initial user input</b> is present as shown in the directions.		20 points
The program uses a <b>Scanner</b> to get user input.		10 points
The code is present to output the <b>starting message</b> using the user's name and the number of problems requested.		15 points
The program uses a <b>while/do while loop</b> to generate the practice problems.		10 points
The program uses a <b>while/do while loop</b> to check for division by 0.		10 points
<b>Neither</b> while loop exits using a <b>break statement</b> .		10 points
The code is present to <b>randomly select two numbers</b> for the practice problems.		20 points
A <b>switch statement</b> is used to set the operation and the answer for each practice problem.		20 points
The code is present to <b>correctly display</b> the practice problems and <b>allow the user to input</b> an answer.		20 points
The code is present to ask for both the <b>quotient and remainder</b> and allow the user to input both if the user chose <b>division</b> .		10 points
<b>if statements</b> are used to check whether the user answered the problem correctly.		10 points
The code is present to provide the <b>appropriate output</b> based on whether the answer is correct.		10 points
The code is present to <b>count</b> the number of <b>correct answers</b> .		10 points
The program uses <b>modulus (%)</b> to find the remainder in division problems.		5 points
The code is present to output the <b>final message</b> using the <b>user's name</b> and the <b>number of problems requested</b> .		10 points
The <b>Scanner</b> was closed.		5 points
<b>Total Points</b>		<b>/515 points</b>