

NAME _____

DATE _____

PERIOD _____

UNIT 4 · TOPIC 4.7 · WORKSHEET

Numbers in Envelopes

Box them, unbox them, parse them, and catch the Integer that refuses to change.

About 30 minutes

9 questions

Cumulative through 4.7

TOPIC

4.7

**DIRECTIONS**

Work each puzzle in pencil. When code prints, write exactly what it prints. An Integer or Double is a number sealed in an object, like a letter in an envelope. Some puzzles include pieces you won't need.

PART A · 3 QUESTIONS

A**Warm up your brain****1**

YOU'RE THE JVM

You are the Java Virtual Machine. Run this code in your head. What prints?

```
String[] parts = "9,4".split(",");
Integer a = Integer.parseInt(parts[0]);
Integer b = Integer.parseInt(parts[1]);
double half = (a + b) / 2;
System.out.println(parts[0] + parts[1] + " " + half);
```

A 94 6.0**B** 13 6.5**C** 94 6.5**D** 13 6.0**THINK FIRST**

Two Integer variables refer to the same object. You add 5 through one of them. Does the other one see the new number? Guess in the margin; question 7 settles it.

2 PENCIL CHECK

The statements run in order. For each one, write whether Java **autoboxes**, **unboxes**, or does **neither**.

Statement	autobox, unbox, or neither?
<code>Integer count = 12;</code>	
<code>int n = count;</code>	
<code>double d = n;</code>	
<code>Double rate = 0.5;</code>	
<code>int big = Math.max(count, 3);</code>	
<code>int parsed = Integer.parseInt("12");</code>	

3 FACE-OFF

Tonight: `int` and `Integer` argue about who's the better number. Fill in the blanks.

INT	I'm simple. My variable holds the number _____. No envelope, no fuss.
INTEGER	I'm an object, so my variable holds a _____. It can even hold _____ when there's no number yet.
INT	Sure, until someone does math with you while you're <code>null</code> . Unboxing you throws a _____.
INTEGER	At least nobody has to import me. I live in the _____ package, the same as <code>String</code> and <code>Math</code> .
INT	And once you're created, your number can never change. You're _____.
INTEGER	Write <code>count = count + 1;</code> and Java just builds a _____ <code>Integer</code> object and points <code>count</code> at it.



WATCH OUT

`parseInt` and `parseDouble` are picky about every character. A file line like `Kim, 14` split on `,` gives the piece `" 14"`, with a space in front, and `parseInt` throws a `NumberFormatException` on that space. Split on `,` instead, so the piece is exactly `"14"`.

PART B · 3 PUZZLES

B Puzzles

4 CODE TILES

A snack bar keeps its prices as text. The program should add them up. Someone dropped the tiles; put them back so the program prints the output below. Two tiles are decoys and aren't used.

```
String[] prices = {"2.50", "4.25", "1.75"};
```

System.out.println("Total: " + total);	total = total + Double.parseDouble(p);
Double total = 0;	for (String p : prices) {
Double total = 0.0;	}
total = total + Integer.parseInt(p);	

Target output

Total: 8.5

YOUR PROGRAM

5 SNIPPET POOL

Hopper's lemonade stand lost money every day this week. Each day's change in dollars arrived as text. Fill each numbered hole with a snippet from the pool so the program finds the *best* day (the largest change) and prints Best day: -15. Each snippet is used at most once.

```
String[] changes = {"-40", "-15", "-22"};
int best = (1);
for (String c : changes) {
    int amount = (2);
    if ((3)) {
        best = amount;
    }
}
System.out.println("Best day: " + best);
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

0	Integer.MIN_VALUE	Integer.MAX_VALUE
Integer.parseInt(c)	Double.parseDouble(c)	c
amount > best	amount < best	



FAIR QUESTION

If an Integer is an object, why didn't anyone write new?

Autoboxing writes it for you. When an `int` lands in an `Integer` variable or parameter, the compiler adds the code that builds the object. Older code sometimes shows `new Integer(12)`, but that constructor is deprecated in current Java and isn't used on the AP exam.

6 YOU'RE THE COMPILER

You are the compiler, and then the JVM. For each snippet, decide whether it compiles. If it doesn't, say why. If it does, write what it prints, or name the exception if it crashes.

Snippet	Compiles?	Output, crash, or why not
1 Double price = 5;		
2 Integer count = 7; double avg = count; System.out.println(avg);		
3 Double rating = 4.0; int stars = rating;		
4 Integer best = null; int copy = best;		
5 int hours = Integer.parseInt("7.5");		

PART C · 3 QUESTIONS

C Catch the mistake

7 TWO TRUTHS AND A LIE

Two of these claims about the code are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
1 Integer stamps = 20;
2 Integer copy = stamps;
3 stamps = stamps + 5;
```

- 1 Right after line 2, stamps and copy refer to the same Integer object.
- 2 After line 3, stamps holds 25.
- 3 Line 3 changes the shared object to 25, so copy holds 25 too.

THE LIE, REWRITTEN

8 WHO AM I?

Some wrapper-class ideas are playing a party game. Each gives a clue about itself. Name it.

Clue	I am...
I'm a class whose objects each hold one <code>int</code> .	
I'm the compiler quietly turning a <code>double</code> into a <code>Double</code> .	
I'm the compiler taking the <code>int</code> out of an <code>Integer</code> for you.	
Give me <code>"75"</code> and I return <code>75</code> . Give me <code>"7.5"</code> and I throw an exception.	
No <code>int</code> is bigger than me, so I'm a safe start when you hunt for the smallest value.	
I'm the package that <code>Integer</code> and <code>Double</code> live in. Nobody imports me.	

9 ONE-LINE FIX

This program prints 10 and 20, but it should print 15 and 30. Change exactly one line of `main` so that `score` really ends up holding 15. Keep the call to `addFive`, and don't change line 7.

Rewards.java

```

1  public class Rewards {
2      public static Integer addFive(Integer points) {
3          points = points + 5;
4          return points;
5      }
6      public static void main(String[] args) {
7          Integer score = 10;
8          addFive(score);
9          System.out.println(score);
10         System.out.println(score * 2);
11     }
12 }
```

WHY IT PRINTS 10, AND YOUR NEW LINE