# Homework M1: Review

**Due** Feb 7 at 11:59pm **Points** 100 **Questions** 39
**Available** until May 30 at 11:59pm **Time Limit** None

# Instructions

Review the **Homework FAQ** page. This page contains a video of how to use the provided tester file. **Make sure you know how to add the tester to your IDE and use it to test your code!** If you aren't sure how to use the tester or have any questions, post to the discussion board!

In this homework, you will:

- trace and evaluate conditionals, loops, methods, and code involving null
- write complete methods using conditionals and loops
- write your own parent and child classes
- follow and evaluate the use of best practices and naming conventions

## Homework Files

I have provided you with a driver/tester program that you can use to test your code before submitting. I **strongly** recommend running this program before you submit your homework. The program is designed to help you catch common errors and fix them before submitting.

The driver program has a series of *test methods* at the end. These are meant to streamline the running of multiple repeated tests. Don't worry too much if the test methods don't make sense- you don't need to do anything with these methods. They are just there to help with the test cases declared in the main method.

**Important Note: The tester will not compile until the Receipt and DigitalReceipt classes exist and have the required method headers.** You can write these classes with placeholder code so the tester compiles. You can also comment out different parts of the tester while working on other parts.

**HomeworkM1Files.zip** (https://ccsf.instructure.com/courses/47904/files/7254588/download?download_frd=1)

This quiz was locked May 30 at 11:59pm.

## Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| **LATEST** | Attempt 1 | 7,653 minutes | 104 out of 100 |

Score for this quiz: **104** out of 100
Submitted Feb 7 at 12:58am
This attempt took 7,653 minutes.

# Coding Questions

In the following questions, write a parent class that describes a **receipt**. A receipt is proof that someone has purchased goods or services. Also write a child class that describes a **digital receipt**. A digital receipt is a special kind of receipt that is emailed to the customer. (This is **inheritance**!)

## The Receipt Class

In your classes, use the Store.java class provided in the **homework files** (https://ccsf.instructure.com/courses/47904/files/7254588/download?download_frd=1) . (This is **aggregation**!)

A receipt is described by:

- the Store where the purchase was made (use the Store class for the type)
- the number of items purchased
- the total amount of money spent
- a receipt ID that could contains letters and numbers

## Digital Receipt

A digital receipt is described by all of the above characteristics and also:

- the email address where the digital receipt is sent

## Writing the Classes

I recommend writing these class in Eclipse (or whatever environment you use) and then copy/pasting the various pieces into the homework. Only paste in the requested method/code for each question- not the entire class.

Also, again, I strongly recommend running the **provided tester/driver program** (https://ccsf.instructure.com/courses/47904/files/7254588/download?download_frd=1) to make sure your classes and methods are working correctly. The **Homework FAQ** page has information and a video that shows how to use the tester program.

The **lecture videos** (https://www.youtube.com/playlist?list=PL5igFWijWBo1D0f-cJHH6jlJujtoggqM_D&disable_polymer=true) for this week review important ideas about class design. There are also several examples that walk through the creation of parent and child classes. Use these examples as a guide.

## For Full Credit:

- Follow Java coding conventions and best practices (discussed in the **Java Coding Conventions lecture video** (https://youtu.be/va7zNueA9N0) )
- Follow good principles of class design, encapsulation, inheritance, and code reuse (discussed in the **Writing Classes** (https://youtu.be/X4W0OlHPEwU) , **Aggregation** (https://youtu.be/X_rd4pjjlkE) , and **Inheritance** (https://youtu.be/hc2AtyegB_8) lecture videos)
- Follow naming conventions for variables, classes, and methods (discussed in the **Java Coding Conventions lecture video** (https://youtu.be/va7zNueA9N0) )
- Reduce duplicated code.
- Properly format your code.

## Question 1 — 5 / 5 pts

Write the complete class header and the instance data variables for the Receipt class.

### For full credit:

- Choose the appropriate type for each the variables
- Declare variables in a way that supports encapsulation
- Follow naming conventions

Your Answer:

```
public class Receipt {

        // instance data variables
        private int numberOfItems;
        private final static int DEFAULT_NUMBER_OF_ITEMS = 1;
        private double total;
        private String receiptID;
        private Store store;
```

## Question 2

**8 / 8 pts**

Write **two** constructors for the Receipt class.

1. the first constructor takes in all four pieces of information as parameters (the Store, number of items, total amount spent, and receipt id)
2. the second constructor takes only the Store, amount spent, and id.
    - in this constructor, use a default value of 1 for the number of items.

If you use any constants, declare them here as part of your answer.

### For full credit:

- Follow best practices related to using constants (discussed in **Variables and Data lecture video (https://youtu.be/Q1vnlfYf54Y)** )
- Follow best practices related to invoking overloaded constructors and reducing duplicated code (discussed in the **Writing Classes lecture video (https://youtu.be/X4W0OlHPEwU)** )

Your Answer:

```
// constructor 1
public Receipt(Store store, int numberOfItems, double total, String receipt
ID) {
        this.store = store;
        this.numberOfItems = numberOfItems > 0 ? numberOfItems : DEFAULT_NU
MBER_OF_ITEMS;
        this.total = total;
        this.receiptID = receiptID;

}

// constructor 2
public Receipt(Store store, double total, String receiptID) {
        this(store, DEFAULT_NUMBER_OF_ITEMS, total, receiptID);
}
```

## Question 3

**2 / 2 pts**

Which of the following would be the best implementation of a setter method for the number of items variable?

```
public void setNumberOfItems(int numberOfItems) {
     if(numberOfItems<0) {
          System.out.println("Error- number of ite
ms cannot be negative.");
     }
     this.numberOfItems = numberOfItems;
}
```

**Correct!**

```
public void setNumberOfItems(int numberOfItems) {
     if(numberOfItems>0) {
          this.numberOfItems = numberOfItems;
     }
}
```

```
public void setNumberOfItems(int numberOfItems) {
     this.numberOfItems = numberOfItems;
}
```

```
public void updateTheNumberOfItemsWithANewValue(in
t numberOfItems) {
     numberOfItems = this.numberOfItems;
}
```

Note: you should write getters and setters for your instance data variables. You won't submit these, but you will need to have them in your class in order for the driver program to run correctly.

## Question 4

**5 / 5 pts**

Write a toString method. The text representation of a receipt should include:

- the text representation of the Store
- the number of items
- the total money spent
- the receipt ID

You are not required to, but if you want to format the money properly, you can use this code to get a formatted String:

```
String moneyString = NumberFormat.getCurrencyInstance
().format(totalMoneySpent);
```

### For full credit:

- The text should be formatted in a way that is human-readable (e.g.,

with spaces, tabs, or new lines).
- Follow good practices of encapsulation. Let each object be in charge of its own functionality (discussed in the **Writing Classes (https://youtu.be/X4W0OlHPEwU)** lecture video).

### +2 Point Optional Extra Credit:

Use the ternary/conditional operator (discussed in the **Conditionals (https://youtu.be/aGm72WjCkwY)** lecture video) to customize whether the returned text includes the 's' based on whether there is more than one item. For example, the text representation would include "1 item" (no 's') but "2 items" (with the 's') or "5 items," etc.

For the full extra credit, write the code that uses the operator in a way that best reduces duplicated code.

Your Answer:

```
// toString METHOD
@Override
public String toString() {
        // logic for item singular or plural
        String itemSingularPlural = numberOfItems > 1 ? "items" : "item";
        // return statement
        return store.toString() + "\nNumber of " + itemSingularPlural + ":
" + numberOfItems
                        + "\nTotal: $" + total + "\nReceipt ID: " + receipt
ID + "\n";
}
```

## Question 5

**8 / 8 pts**

Override the equals method inherited from Object.

Two receipts should be defined to be *the same* (logically equivalent) if each of the following is true:

- the Stores are logically equivalent
- the number of items is the same
- the total money spent is the same
- the receipt IDs are the same ignoring case/capitals (e.g., "abc" is

considered equivalent to "ABC")

### For full credit:

- Follow good practices of encapsulation. Let each object be in charge of its own functionality. These ideas are discussed in the **equals Method (https://youtu.be/gaj93xco_p4)** lecture video and **example (https://youtu.be/e8r8W2xbLY4)** video that follows that one.

Your Answer:

```
// equals METHOD
@Override
public boolean equals(Object obj) {
        if (obj instanceof Receipt) {
                // cast obj to Receipt class
                Receipt otherReceipt = (Receipt)obj;

                if(this.store.equals(otherReceipt.store) && this.total == o
therReceipt.getTotal() &&
                                this.receiptID.equalsIgnoreCase(otherReceip
t.getReceiptID())
                                && this.numberOfItems == otherReceipt.getNu
mberOfItems()){
                        return true;
                }
        }
        return false;
}
```

note: because of precision and rounding, avoid comparing doubles (and floats) directly with ==; instead, use a setup like this: Math.abs(d1-d2) < 0.00001; post a comment to the discussion board to see an example of == not returning the correct result

Assume that a receipt id contains characters that represent information or that can be used as a code to determine something about the purchase. For example: if an id has an 'x' followed by two 'y's, it means that the purchase was a final sale and cannot be returned. Or if an id has an 'a' followed by three 'b's, it means that a coupon was used for purchase.

Write a static method for the Receipt class to test whether a receipt id meets such a criteria.

## Does the id meet the criteria?

The criteria is described by three pieces of data, passed in as parameters:

- char firstTarget
- char secondTarget
- int countOfSecondAfterFirst

A receipt id meets the criteria if:

- the id contains the first target character **and**
- the second target character appears **exactly** the specified number of times **after** the first target character appears

Note: the first and second char could be the same char!

Carefully review the provided driver program for examples of ids that meet and don't meet criteria.

## For full credit

Do no import any classes from the java.util.regex package. Do not use regular expressions. (This isn't what I want you to practice in this question!)

## Method Header

The method header in the Receipt class is:

```
public static boolean idMeetsCriteria(String receiptI
D, char firstTarget, char secondTarget, int countOfSec
ondAfterFirst)
```

## Question 6

**10 / 10 pts**

Paste your complete method here.

Your Answer:

```
// receiptId criteria method
public static boolean idMeetsCriteria(String receiptID, char firstTarget, c
har secondTarget,
                int countOfSecondAfterFirst) {
        // booleans to check criteria
        boolean idMeetsCriteria1 = false;
        boolean idMeetsCriteria2 = false;
        boolean idMeetsCriteria3 = false;

        // checking if id contains firstTarget char
        if (receiptID.indexOf(firstTarget) != -1) {
                idMeetsCriteria1 = true;
        }

        // checking if id contains secondTarget char after the firstTarget
car
        if (idMeetsCriteria1 == true) {
                int indexOfFirstTarget = receiptID.indexOf(firstTarget);
                String stringAfterFirstTarget = receiptID.substring(indexOf
FirstTarget + 1);

                if (stringAfterFirstTarget.indexOf(secondTarget) != -1  ||
countOfSecondAfterFirst == 0) {
                        idMeetsCriteria2 = true;
                }

                //loop to check if secondTarget occurs valid number of time
s
                int secondCharCounter = 0;

                if (idMeetsCriteria2 == true) {
                        for (int i = 0; i < stringAfterFirstTarget.length
(); i++) {
                                if(stringAfterFirstTarget.charAt(i) == seco
ndTarget) {
                                        secondCharCounter++;
                                }
                        }
                        if (countOfSecondAfterFirst == secondCharCounter) {
                                idMeetsCriteria3 = true;
                        }
                }
        }

                // returns true if all criteria are met
                if(idMeetsCriteria1 && idMeetsCriteria2 && idMeetsCriteria
3) {
                        return true;
                }

        return false;
}
```

## Question 7

**8 / 8 pts**

Write the following for the **child class** called DigitalReceipt:

1. the complete child class header
2. the instance data variables
3. one constructor that takes in all five pieces of information that describe a digital receipt

### For full credit:

Follow best practices related to inheritance and code reuse (discussed in the **Inheritance** (https://youtu.be/hc2AtyegB_8) lecture video and the **example** (https://youtu.be/0wU-pNxEP8U) that follows that video).

Your Answer:

```
public class DigitalReceipt extends Receipt{

        // instance data variables
        private String email;

        // constructor 1
        public DigitalReceipt(Store store, int numberOfItems, double total,
String receiptID, String email){
                super(store, numberOfItems, total, receiptID);
                this.email = email;

        }
}
```

## Optional Extra Credit (10 points)

Write a complete static method to determine if an email is valid. This method will be placed inside the DigitalReceipt class.

For this question, an email is valid if **all** of the following is true:

- it contains the @ symbol with at least one **letter** character (i.e., a-z or A-Z) at *some* position somewhere before the @

- it contains at least one period (.)
- there are **exactly** three characters after the **last** period

Review the driver program for examples of valid and invalid addresses.

### For full credit:

- Do not import any classes from the java.util.regex package. (Do not use regular expressions.)
- Submit the complete method with method header.
- The method should be static.
- Use existing methods in the String and Character classes when possible, instead of re-writing code yourself.

### Method Header

The method header in the DigitalReceipt class is:

```
public static boolean validateEmail(String email)
```

## Question 8

**7 / 0 pts**

Optional Extra Credit: Paste your complete method here.

Your Answer:

```
// validate email method
public static boolean validateEmail(String email) {

	boolean validEmail = true;
	int indexAt = email.indexOf("@");
	int indexDot = email.indexOf(".");

	// check for @
	if(indexAt == -1) {
		return false;
	}

	// check for "."
	if(indexDot == -1) {
		return false;
	}

	if (indexAt > -1) {
		String stringBeforeAt = email.substring(0, indexAt);
```

```
                // check for char before @
                if(stringBeforeAt.length() < 1) {
                        return false;
                }
        }

        if (indexDot > 0) {
                String stringAfterDot = email.substring(indexDot);
                // check for 3 chars after period
                if(stringAfterDot.length() > 4 || stringAfterDot.length() <
4){
                        return false;
                }
        }

        return true;
}
```

-3 use the lastIndexOf the period, instead of the indexOf; the code doesn't return the correct result if there are no *letters* before the @ (e.g., email=!@abc.def should return false but it returns true)

# Multiple Choice and Short Answer Questions

## Question 9

**3 / 4 pts**

What best describes each of the following **variable** names?

- **valid, follows convention**: the variable name is allowed by the compiler and also follows appropriate naming conventions for Java variables
- **valid, does not follow convention**: the variable name is valid- it is allowed by the compiler- but it does not follow naming conventions

for Java variables

- **not valid**: the variable name is not allowed by the compiler

Note that this question asks about Java **variables** (**not** constants or classes). You can review naming conventions in the **Java Coding Conventions** (https://youtu.be/va7zNueA9N0) lecture video.

| | | |
|---|---|---|
| Correct! | **total-value** | not valid |
| Correct! | **total_value** | valid, does NOT follow cc |
| Correct! | **totalValue** | valid, follows convention |
| Correct! | **totalvalue** | valid, does NOT follow cc |
| Correct! | **Value** | valid, does NOT follow cc |
| Correct! | **TotalValue** | valid, does NOT follow cc |
| ou Answered | **tv** | valid, follows convention |
| | **Correct Answer** | valid, does NOT follow convention |
| ou Answered | **TOTAL_VALUE** | valid, follows convention |
| | **Correct Answer** | valid, does NOT follow convention |

## Question 10

1 / 1 pts

What is the value of expression?

```
boolean done = !false;
int x = 11, y = 11;
boolean expression = (done && x<=y);
```

Correct!

- (•) True
- ( ) False

## Question 11

1 / 1 pts

What is the value of expression?

```
boolean a = true;
boolean b = true;
boolean expression = a || !b;
```

Correct!

- (•) True
- ( ) False

*****You will need this keyword for a question in Homework M2: eureka

## Question 12

1 / 1 pts

What is the value of expression?

```
boolean a = true;
boolean b = true;
boolean c = false;
boolean expression = !(b || c) && a;
```

- True
- False

Correct!

## Question 13

2 / 2 pts

What is the output of the following code?

```
int num = 50;

if(num >= 30) {
   System.out.print(1);
} else if(num >= 40) {
   System.out.print(2);
} else if(num >= 50) {
   System.out.print(3);
} else {
   System.out.print(4);
}
System.out.print(5);
```

Correct!

- 15
- 2

- 12345
- 125
- 5
- none of these is correct
- 25
- 1
- 2345

## Question 14

**2 / 2 pts**

What is the output of the following code?

```
int a = 70;
boolean b = false;

if(a >= 70) {
   System.out.print(1);

   if(b==true) {
      System.out.print(2);
   }

} else {
   System.out.print(3);

   if(b==false) {
      System.out.print(4);
   }

}
System.out.print(5);
```

**Correct!**

- (•) 15
- ( ) 14
- ( ) none of these is correct
- ( ) 12345
- ( ) 12
- ( ) 145
- ( ) 125
- ( ) 1345

## Question 15

**2 / 2 pts**

What is the output of the code above using these initial values?

```
int a = 43;
boolean b = false;
```

- ( ) 12345
- ( ) 35
- ( ) 45
- ( ) 1234
- ( ) 5
- ( ) 3

- none of these is correct

Correct!

- (selected) 345
- 34

## Question 16

**1 / 1 pts**

The following method is **syntactically** correct (meaning it will compile).

```
public boolean method() {
   int value = 5;
   if(value > 0) {
      return true;
   }
}
```

- True

Correct!

- (selected) False

The compiler does not evaluate boolean expressions in conditionals; these are evaluated at runtime. The compiler only knows that it's possible for a single if-statement not to execute. If that happens, there will be no return statement. So the code will not compile.

## Question 17

**0 / 1 pts**

The following method is **syntactically** correct (meaning it will compile).

```
public boolean method() {
   int value = 5;
   if(value > 0) {
      return true;
   } else {
      return false;
   }
}
```

orrect Answer

- True

ou Answered

- False

With an if-else statement, the compiler knows that exactly one branch will execute. Since there is a return statement in both branches, the code will compile.

## Question 18

**2 / 2 pts**

Suppose you want to write code to print information about whether a number is even **and** whether the number is divisible by 3.

Examples:

- for the number 4, the code will print "The number is even"
- for the number 9, the code will print "The number is divisible by three"
- for the number 6, the code will print "The number is even" **and** "The number is divisible by three"

Which of the following is the **best** solution?

Note that more than one answer might be correct to accomplish the task. Select the **best** solution. Select the conditional that best matches the required task.

**Correct!**

```
if(number % 2 ==0 ) {
   System.out.println("The number is even.");
}
if(number % 3 == 0) {
   System.out.println("The number is divisible by
three.");
}
```

```
if(number % 2 ==0 ) {
   System.out.println("The number is even.");
} else if (number % 3 == 0) {
   System.out.println("The number is divisible by
three.");
}
```

```
if(number % 2 ==0 ) {
   System.out.println("The number is even.");
} else {
   System.out.println("The number is divisible by
three.");
}
```

none of these will correctly accomplish the task

Both conditions could be true, so they need to be checked with separate conditional blocks.

## Question 19

**2 / 2 pts**

Suppose you want to write code to print a statement about whether a number is in one of three ranges: low, medium, or high. Which of the

following is the **best** solution?

Note that more than one answer might be correct to accomplish the task. Select the **best** solution. Select the conditional that best matches the required task.

```
if(number<low) {
   System.out.println("The number is low.");
}
if (number>=low && number<=high) {
  System.out.println("The number is in the middl
e.");
} else {
   System.out.println("The number is high.");
}
```

```
if(number<low) {
   System.out.println("The number is low.");
   if (number>=low && number<=high) {
      System.out.println("The number is in the mid
dle.");
   } else if(number>high) {
      System.out.println("The number is high.");
   }
}
```

**Correct!**

```
if(number<low) {
   System.out.println("The number is low.");
} else if(number>=low && number<=high) {
   System.out.println("The number is in the middl
e.");
} else { // (number>high)
   System.out.println("The number is high.");
}
```

```
if(number<low) {
   System.out.println("The number is low.");
}
if (number>=low && number<=high) {
   System.out.println("The number is in the middl
e.");
}
if(number>high) {
   System.out.println("The number is high.");
}
```

The linked if-else if is the best approach. The conditions are exclusive- meaning only one will be true. Thus, once you find a true condition, you don't want to check the others.

## Question 20

**1 / 1 pts**

Consider the following code that will assign a letter grade of 'A', 'B', 'C', 'D', or 'F' depending on a student's test score. Is the correct letter (a 'B') assigned for a grade of 85?

```
if (score >= 90) {
   grade = 'A';
}
if (score >= 80)  {
   grade = 'B';
}
if (score >= 70) {
   grade = 'C';
}
if (score >= 60) {
   grade = 'D';
}
else {
   grade = 'F';
}
```

Correct!

- (●) No, the correct letter is not assigned.
- ( ) Yes, the correct letter is assigned.

## Question 21

**2 / 2 pts**

What is the value of **sum** after the code is executed?

```
int i=7;
int sum = 0;
while(i%5 > 0) {
     sum += i;
     i++;
}
```

Correct!

24

orrect Answers

24 (with margin: 0)

## Question 22

**2 / 2 pts**

What is the value of **i** after the code above is executed?

Correct!

10

orrect Answers

10 (with margin: 0)

## Question 23

**0 / 1 pts**

The following code is **syntactically** correct (meaning it will compile).

```
for (int k=0; k<10; k++) {
   System.out.println(k);
}
System.out.println("final value of k is " + k);
```

ou Answered

- (•) True

rrect Answer

- ( ) False

variables declared inside of the for-loop are **not** visible outside of the for-loop

## Question 24

1 / 1 pts

The following code is **syntactically** correct (meaning it will compile). Assume you have an object called scan that reads input from the user.

```
int value;
while(value < 0) {
   value = Integer.parseInt(scan.nextLine());
}
System.out.println("Your non-negative number is " + val
ue);
```

- ( ) True

Correct!

- (•) False

Because a while loop might never be executed, a variable that is declared outside the loop and only initialized inside the loop cannot be used outside the loop because there is a chance it will never have been initialized.

## Question 25

**2 / 2 pts**

What is the value of **sum** after the code is executed?

```
int i=0;
int sum = 0;
for(i=0; i<5; i++) {
     sum += i;
}
```

**Correct!**

10

**Correct Answers**

10 (with margin: 0)

## Question 26

**2 / 2 pts**

What is the value of **i** after the code above is executed?

**Correct!**

5

**Correct Answers**

5 (with margin: 0)

## Question 27

2 / 2 pts

How many times is "hello" printed?

```
for(int i=0; i<10; i++) {
     for(int j=0; j<5; j++) {
          System.out.println("hello");
    }
}
```

Correct!

50

orrect Answers

50 (with margin: 0)

## Question 28

1 / 1 pts

The following code is **syntactically** correct (meaning it will compile).

```
public int method(int num) {
   for(int i=0; i<num; i++) {
      System.out.println(i);
      return num;
   }
}
```

- ○ True

Correct!

- ◉ False

The compiler only knows that a for loop might not execute. If that happens, there is no return statement outside the for-loop. So the code will not compile.

## Question 29

**1 / 1 pts**

The following code will throw an exception (crash).

```
Student s = null;
if(s != null && s.hasPaidTuition()) {
     System.out.println("paid");
}
```

- ○ True

**Correct!**

- ◉ False

Java conditionals use short-circuit execution. This means if the value of the entire expression can be determined by the left-hand operand, the right-hand operand is never evaluated. Since s!=null is false, s.hasPaidTuition() is never invoked, so no exception is thrown.

## Question 30

**2 / 2 pts**

What is the result of each of the following statements using the variables below?

```
String s = "";
String n = null;
```

Correct! **System.out.println(s.length** — 0 is printed

Correct! **System.out.println(n.length** — an exception or error is tl

Other Incorrect Match Options:

- -1 is printed
- none of these is correct
- false is printed
- true is printed

## Question 31

**4 / 4 pts**

What is the result of each of the following statements using the variables below?

```
String a = null;
String b = "";
```

Correct! **System.out.println(a==b);** — false is printed

Correct! **System.out.println(a.equals** — an exception or error is tl

Correct! **System.out.println(b==a);** — false is printed

Correct! **System.out.println(b.equals** — false is printed

Other Incorrect Match Options:

- true is printed
- none of these is correct

## Question 32

**2 / 2 pts**

Consider the code fragments below:

```
Person person = new Person(...);

int number = 4;
```

What is stored in the memory location of each variable:

Correct! **person** — a memory location to son

Correct! **number** — the value 4

Other Incorrect Match Options:

- the information that describes a person
- something else not listed here

## Question 33

**2 / 2 pts**

What is the output of the code snippet below?

```
int n = 10;
method(n);
System.out.println(n);

public static void method(int n) {
     n++;
}
```

Correct!

10

orrect Answers

10 (with margin: 0)

## Question 34

2 / 2 pts

What is the output of the code snippet below?

Assume there is a class called Employee with an instance data variable (and setter) for ID.

```
Employee e = new Employee(10);
method(e);
System.out.println(e.getId());

public static void method(Employee e) {
     e.setId(99);
}
```

Correct!

99

orrect Answers

99 (with margin: 0)

## Question 35

2 / 2 pts

What is the output of the code snippet below?

```
int[] numbers = {0, 0, 0};
method(numbers);
System.out.println(numbers[1]);


public static void method(int[] numbers) {
     numbers[1] = 43;
}
```

**Correct!**

43

**orrect Answers**

43 (with margin: 0)

## Question 36

**2 / 2 pts**

What is the output of the code snippet below?

```
int[] numbers = {0, 0, 0};
method(numbers);
System.out.println(numbers[1]);


public static void method(int[] numbers) {
     numbers = new int[3];
     numbers[0] = 43;
     numbers[1] = 43;
     numbers[2] = 43;
}
```

**Correct!**

0

**orrect Answers**

0 (with margin: 0)

## Question 37

**2 / 2 pts**

Consider the partial classes shown below.

```
public class Parent {
   private int value;

   ... // full class not shown

   public int getValue() {
      return value;
   }
   public String toString() {
      return "Value=" + value;
   }
}

public class Child extends Parent {
   private String name;

   ... // full class not shown

}
```

Which of the following is the best implementation of the Child class toString method?

**Correct!**

- ◉ `return super.toString() + "\tName=" + name;`
- ○ `return "Value=" + super.getValue() + "\tName=" + name;`
- return "ID: " + getId() + "\nSignature Required: " + signatureRequired;
- ○ `return "Value=" + getValue() +"\tName=" + name;`
- ○ `return this.toString() + "\tName=" + name;`

```
return "Value=" + value + "\tName=" + name;
```

For the following questions, assume you have these classes:

```
public class Parent
public class Child extends Parent
```

## Question 38

**1 / 1 pts**

The following statement is allowed.

```
Parent p = new Child();
```

**Correct!**

- (•) True
- ( ) False

## Question 39

**1 / 1 pts**

The following statement is allowed.

```
Child c = new Parent();
```

- ( ) True

**Correct!**

- False

Quiz Score: **104** out of 100