COMPSCI 121: CLASSES & METHODS CONTINUED

SPRING 20

GOALS FOR TODAY'S CLASS

Methods Review: zyBooks 9.1 to 9.8

Also see lecture slides from Weeks 2 and 3 for:

- Constructors
- Mutators and accessors
- Objects and References

Today

- Coding a program to specification - example.
- Please download the starter code from Moodle and open in jGRASP.

IN-CLASS PROJECT: RESTAURANT APP

Initial project description:

Create an app that stores a list of restaurants. Each restaurant has a `name`, a `type`, and a `rating`. The possible types are: `Fast Food`, `Bistro`, `Fine Dining`. Ratings are `1` to `5`.

Design: Model each restaurant with the `Restaurant` class. Another class, `RestaurantMain`, will handle user input and manage the list of restaurants. The list of restaurants will be an array of `Restaurant` objects.

IN-CLASS PROJECT: RESTAURANT APP

Design time:

Define data types and any constraints on `Restaurant` attributes. Constraints can be a specific length or range of allowable values.

- `name`
- `type`
- `rating`

What are data types and constraints for these attributes?

CHOOSING DATA TYPES AND CONSTRAINTS

1. First understand the initial description of what the app should do.
2. Then, use your knowledge of Java to choose a data type.
3. Finally, add any constraints to reduce the possibility of errors/bad data.

Example: the `rating` attribute:

Data type choice depends on how the app does the ratings:

- `double` if real-valued calculations, such as average are required.
- `int` if a count or simple operations such as add are required.
- `String` if discrete categories required.
- Constraint: it was stated that values are in 1 to 5 inclusive.

IN-CLASS PROJECT: RESTAURANT APP

Design solution:

- `name:` `String` (has no practical constraints).
- `type:` `String`, one of "`Fast Food`", "`Bistro`", "`Fine Dining`" (should be one of the values in the initial description).
- `rating`: an int with range 1:5 inclusive (will only be added to, so integer.

TODO 1 : DECLARE INSTANCE VARIABLES

Implementation time:
The `Restaurant` and `RestaurantMain` class definitions are provided. The code for `RestaurantMain` is also provided.

You will implement the `Restaurant` class code. Please follow the starter code (download from Moodle).

```
//TODO 1: Declare instance variables
for name, type and rating.
```

SOLUTION 1: DECLARING INSTANCE VARIABLES

```
  //TODO 1: Declare instance variables for
name, type and rating.
    private String name;
    private String type;
    private int rating;
```

Q. Why should the variables be declared as `private`?

TODO 2: WRITING A DEFAULT CONSTRUCTOR

How to choose default values:

- A default constructor takes no parameters. Therefore, if we create a new `Restaurant` object using that constructor we don’t have values for `name`, `type`, and `rating`.
- We want default values that indicate the fact that the “real” values are not yet known. These are “sentinel” values.

A “sentinel“ value is a value that is not in the normal or expected range.
It signifies an exceptional condition- in this case that range has not been determined yet. What could these be for a String and an int variable?

TODO 2: WRITING A DEFAULT CONSTRUCTOR

Write a default constructor for the `Restaurant` class that initializes all instance variables to default values.

Initialize `name` and `type` to the empty `String`.

Initialize `rating` to 0. This is not a valid rating, so it signals that rating has not been determined yet.

Why these values?

The empty String, "", and 0 are "sentinel" values- a value that is not in the normal range of specified values.
It signifies an exceptional condition- in this case that the values for name, type, and range have not been determined yet.

TODO 2: WRITING A DEFAULT CONSTRUCTOR

Write a **default** constructor for the `Restaurant` class that initializes all instance variables to **default** values.

Reminder: Default values are *initial* or *starting* values.

TODO 2 SOLUTION: DEFAULT CONSTRUCTOR

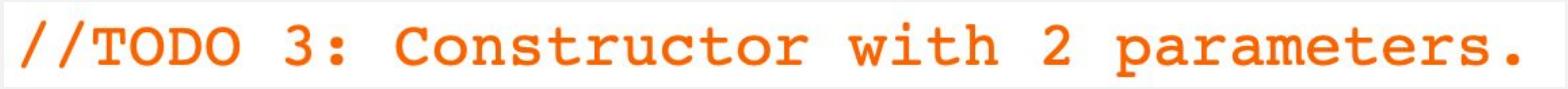
```
//TODO 2: Default constructor
  public Restaurant() {
      name = "";
      type = "";
      rating = 0;
  }
```

Q. Why are these sentinel values?

TODO 3: WRITING A CONSTRUCTOR WITH 2 PARAMETERS

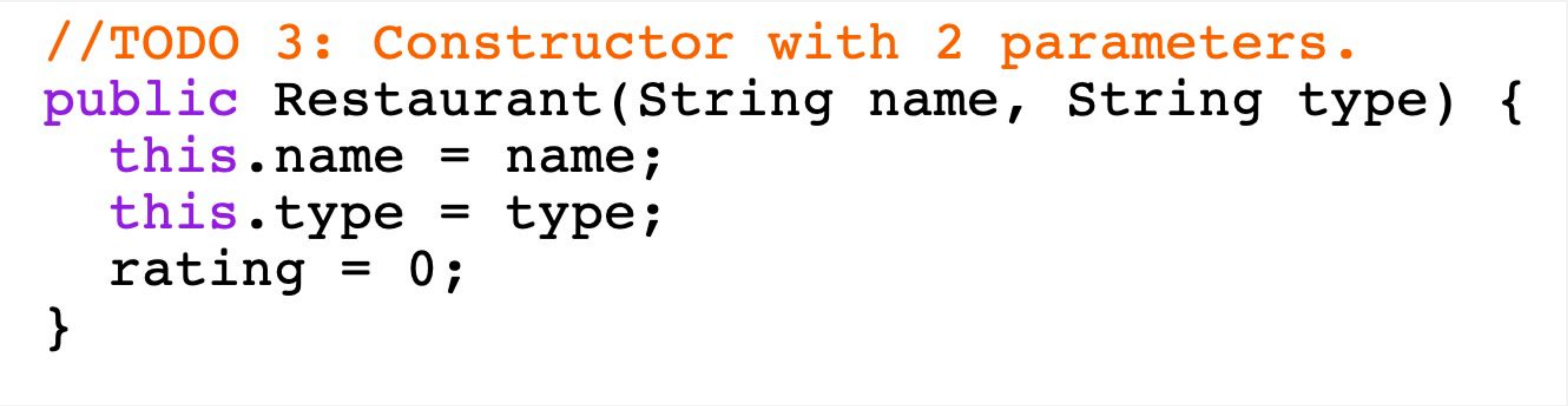
Write a constructor for the `Restaurant` class that initializes the `name` and `type` values to its arguments. It initializes `rating` to the default value.

This means that a `Restaurant` object can be created when we know the `name` and the `type`, but not the `rating`.



```
//TODO 3: Constructor with 2 parameters.
```

TODO 3 SOLUTION: CONSTRUCTOR WITH 2 PARAMETERS



```
//TODO 3: Constructor with 2 parameters.
public Restaurant(String name, String type) {
  this.name = name;
  this.type = type;
  rating = 0;
}
```

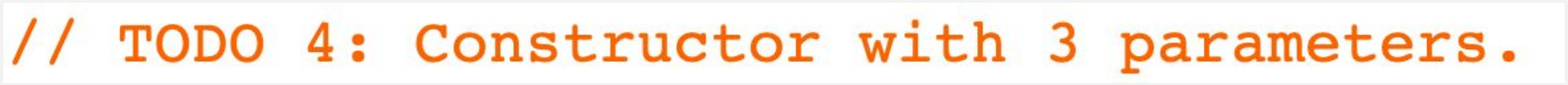
Q. Why use the keyword "`this`"?

TODO 4: WRITING A 3 PARAM CONSTRUCTOR

Write a constructor for the `Restaurant` class that initializes the `name`, `type` and `rating` values to its arguments.

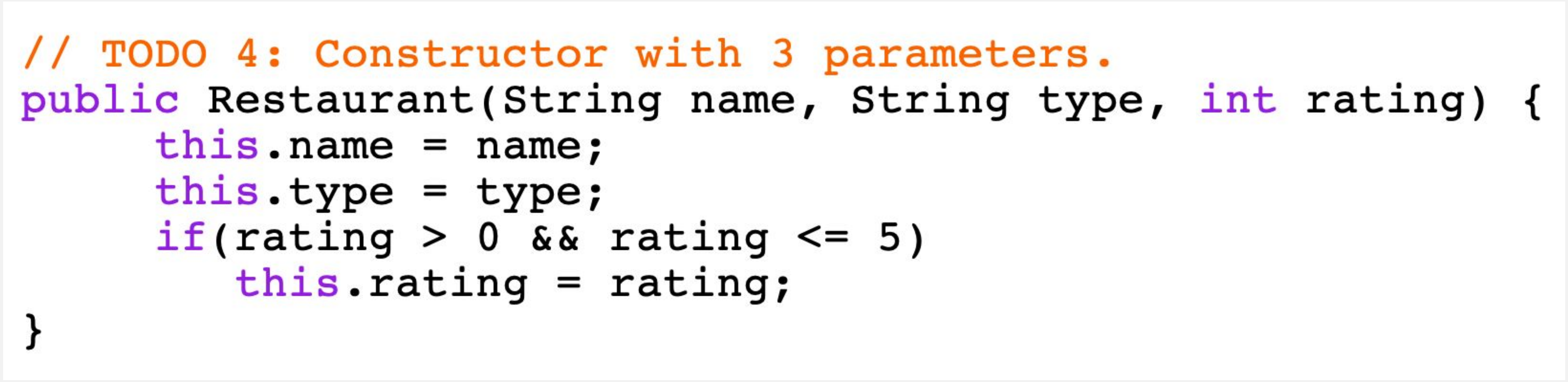
This means that a `Restaurant` object can be created when we know the values of all three attributes.

NOTE: You have to check that the rating passed in is in the range 1:5!



```
// TODO 4: Constructor with 3 parameters.
```

TODO 4 SOLUTION: 3 PARAM CONSTRUCTOR

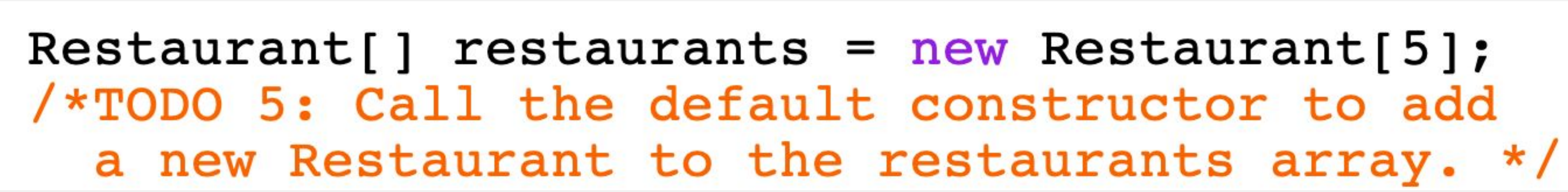


```java
// TODO 4: Constructor with 3 parameters.
public Restaurant(String name, String type, int rating) {
     this.name = name;
     this.type = type;
     if(rating > 0 && rating <= 5)
        this.rating = rating;
}
```

NOTE:
Constructor overloading is the definition of more than one version of the constructor.
Each version must have a different parameter list.

TODO 5: CALLING THE DEFAULT CONSTRUCTOR

In the `RestaurantMain` class, a list of restaurants has been created:



```
Restaurant[] restaurants = new Restaurant[5];
/*TODO 5: Call the default constructor to add
   a new Restaurant to the restaurants array. */
```

In TODO 5, where would you add the new instance in the array?

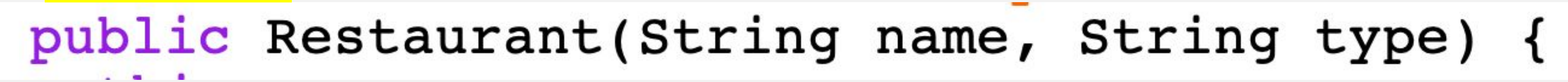
TODO 5: SOLUTION: CALLING THE DEFAULT CONSTRUCTOR

```
 Restaurant[] restaurants = new
Restaurant[5];
/*TODO 5: Call the default
constructor to add new Restaurant to
the restaurants array. */
restaurants[0] = new Restaurant();
```

TODO 6 and 7: CALL THE OTHER CONSTRUCTORS

TODO 6: Write a statement that creates a new `Restaurant` object with the values name "`Serenas`" and type "`Fine Dining`". Assign this object to the second cell in the restaurants array.

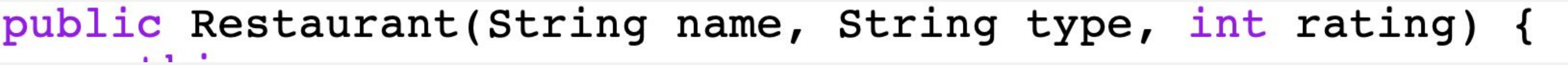
Constructor



```
public Restaurant(String name, String type) {
```

TODO 7: Write a statement that creates a new `Restaurant` object with the values name "`Sams`", type "`Breakfast`", and rating `3`. Assign this object to the third cell in the restaurants array.

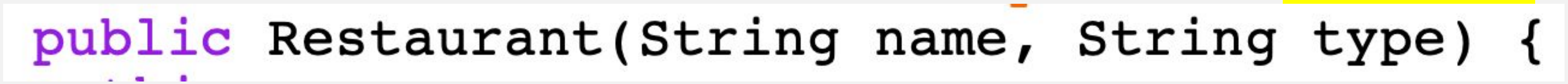
Constructor



```
public Restaurant(String name, String type, int rating) {
```

TODO 6: SOLUTION

Constructor

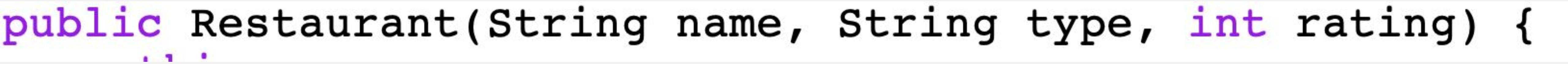


```
public Restaurant(String name, String type) {
```

```
/*TODO 6: Call the 2 parameter
constructor to add a new Restaurant
to the restaurants array. */
restaurants[1] = new
Restaurant("Serenas", "Fine Dining");
```

TODO 7: SOLUTION

Constructor



```java
public Restaurant(String name, String type, int rating) {
```

```java
/*TODO 7: Call the 3 parameter
constructor to add a new Restaurant
to the restaurants array. */
restaurants[2] = new
Restaurant("Sams", "Breakfast", 3);
```

OVERLOADED CONSTRUCTOR PROBLEM

Consider this call:

```
restaurants[1] = new
Restaurant("Serenas", "Fine Dining");
```

Q. What happens if we switch the positions of the parameters?

```
 restaurants[1] = new Restaurant("Fine
Dining", "Serenas");
```

Watch out for this problem!!

TODO 8 and 9: WRITE USEFUL PUBLIC METHODS

TODO 8: Write the `incrementRating` **method** that adds 1 to the current rating as long at the rating will not go above 5.

```
/* TODO 8: The incrementRating method increments the rating by one.
 * It ensures that the rating does not exceed the max of 5.
*/
```

TODO 9: Write the `decrementRating` **method** that subtracts 1 from the current rating as long at the rating will not go below 1.

```
/* TODO 9: The decrementRating method decrements the rating by one.
 * It ensures that the rating does not fall below the min of 1.
*/
```

Consider: Method name (given) and visibility? Return type? Parameter/s?

TODO 8 SOLUTION

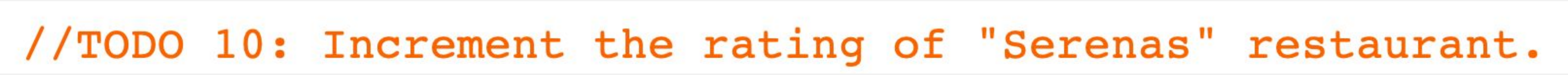
```
/* TODO 8: The incrementRating method
increments the rating by one.
 * It ensures that the rating does not
exceed the max of 5.
*/
   public void incrementRating() {
      if(rating + 1 <= 5)
         rating += 1;
   }
```

TODO 9 SOLUTION

```
/* TODO 9: The decrementRating method
decrements the rating by one.
 * It ensures that the rating does not
fall below the min of 1.
*/
   public void decrementRating() {
      if(rating - 1 > 0)
         rating -= 1;
   }
```

TODO 10: CALL A PUBLIC METHOD

TODO 10: In `RestaurantMain`, write a statement that increments the rating of "`Serenas`".



```
//TODO 10: Increment the rating of "Serenas" restaurant.
```

Use the correct location in the `restaurants` array!

What public method do you call?

TODO 10 SOLUTION: CALL A PUBLIC METHOD

```
//TODO 10: Increment the rating of
"Serenas" restaurant.
        restaurants[1].incrementRating();
```

Use the correct location in the restaurants array!

What public method to call? `incrementRating()`

Clicker Question 1:

```
public Restaurant(String name, String type) {
   this.name = name;
   this.type = type;
   rating = 0;
}
public class RestaurantFavorites {
   public static void main(String[] args) {
      Restaurant foodPlace = new Restaurant();
      ...
   }
}
```

Assume for this question there is no other constructor.

What happens when this code is run?

A. Error, there is no default constructor.
B. The new `foodPlace` instance is created as Java provides the default constructor.
C. The file compiles but the values of the variables are not initialized.

Clicker Question 1: Answer

```
public Restaurant(String name, String type) {
   this.name = name;
   this.type = type;
   rating = 0;
}
public class RestaurantFavorites {
   public static void main(String[] args) {
      Restaurant foodPlace = new Restaurant();
      ...
   }
}
```

Assume for this question there is no other constructor.

What happens when this code is run?

A. Error, there is no default constructor.
B. The new `foodPlace` instance is created as Java provides the default constructor.
C. The file compiles but the values of the variables are not initialized.

Clicker Question 2

```
public class Restaurant {
    public Restaurant(String name, String type) {
   …
```

What happens when this code is run?

```
public class RestaurantFavorites {
    public static void main(String[] args) {
   Restaurant foodPlace = new Restaurant("Taco Heaven",
                                                "Bistro", 4);
```

A. The new `foodPlace` instance is created.
B. The file compiles but the values of the variables are not initialized.
C. Error, the number of parameters do not match the constructor.

Clicker Question 2: Answer

```
public class Restaurant {
    public Restaurant(String name, String type) {
   …
```

What happens when this code is run?

```
public class RestaurantFavorites {
    public static void main(String[] args) {
   Restaurant foodPlace = new Restaurant("Taco Heaven",
                                                  "Bistro", 4);
```

A. The new `foodPlace` instance is created.
B. The file compiles but the values of the variables are not initialized.
C. Error, the number of parameters do not match the constructor.

Clicker Question 3

What happens when this code is run?

```java
public Restaurant(String initName, int initRating) {
    rating = initRating;
    name = initName;
}
```

With this constructor call

Assume for this question there is no other constructor.

```java
Restaurant beveragePlace = new Restaurant(4, "Sam's Beverages");
beveragePlace.print();
```

A. The new `beveragePlace` instance is created.
B. The file compiles but the values of the variables are not initialized.
C. Error, order of parameters is wrong.

Clicker Question 3: Answer

What happens when this code is run?

```
public Restaurant(String initName, int initRating) {
    rating = initRating;
    name = initName;
}
```

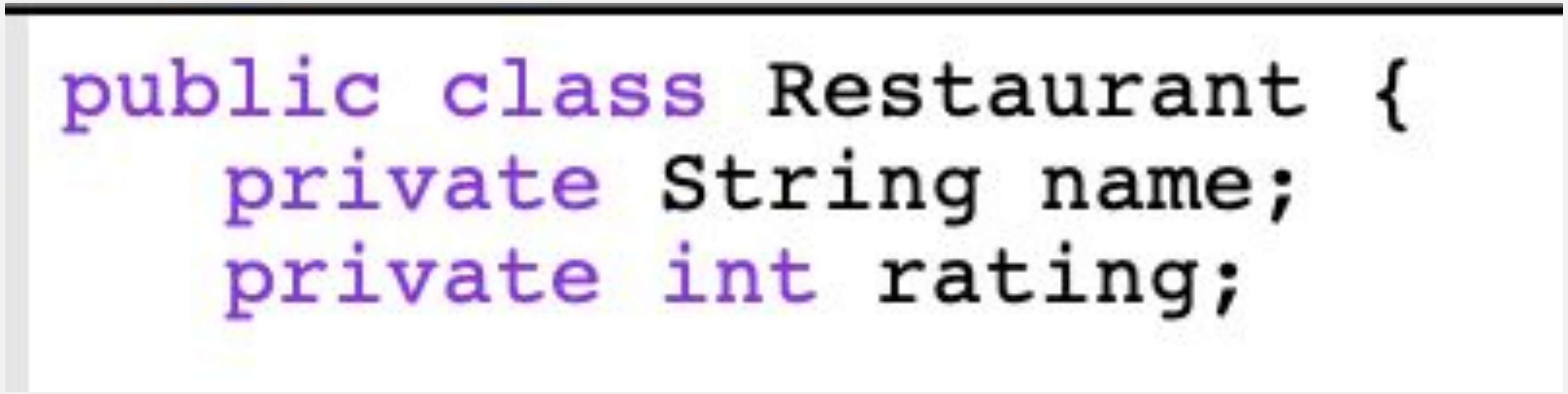
With this constructor call

Assume for this question there is no other constructor.

```
Restaurant beveragePlace = new Restaurant(4, "Sam's Beverages");
beveragePlace.print();
```

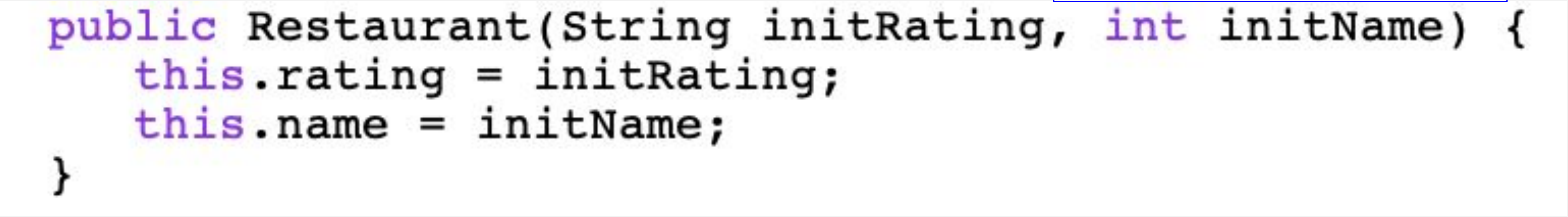
A. The new `beveragePlace` instance is created.
B. The file compiles but the values of the variables are not initialized.
C. Error, order of parameters is wrong.

Clicker Question 4



```java
public class Restaurant {
    private String name;
    private int rating;
```

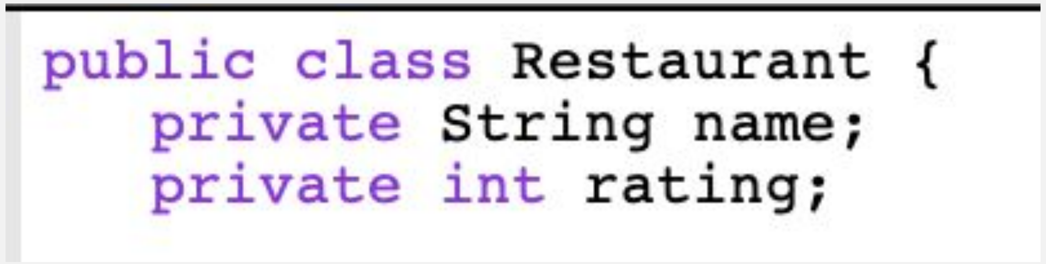
What happens when the constructor is called?



```java
public Restaurant(String initRating, int initName) {
    this.rating = initRating;
    this.name = initName;
}
```

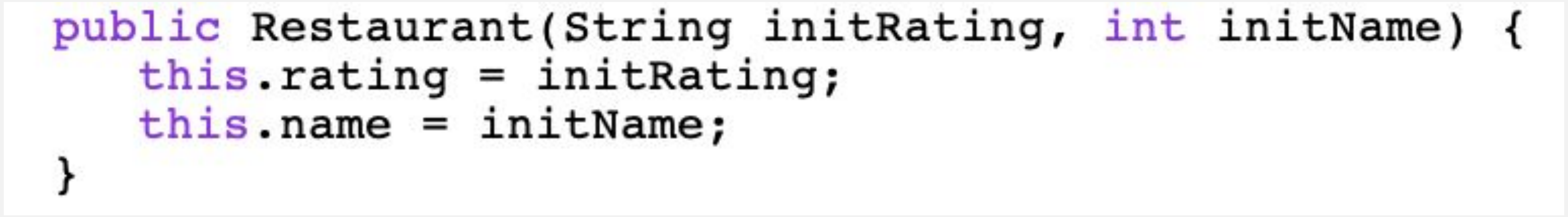
A. Error: incompatible type.
B. The file compiles and the values of the variables are initialized.
C. Error: incorrect assignment statements.

Clicker Question 4: Answer

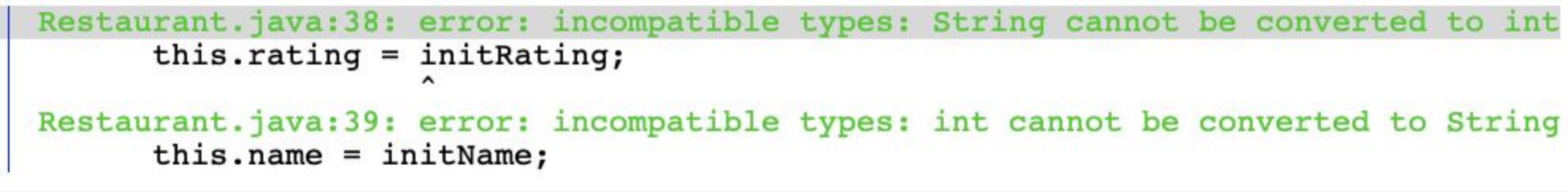


```
public class Restaurant {
   private String name;
   private int rating;
```

What happens when the constructor is called?



```
public Restaurant(String initRating, int initName) {
    this.rating = initRating;
    this.name = initName;
}
```



```
Restaurant.java:38: error: incompatible types: String cannot be converted to int
        this.rating = initRating;
                      ^
Restaurant.java:39: error: incompatible types: int cannot be converted to String
        this.name = initName;
```

A. Error: incompatible type.
B. The file compiles and the values of the variables are initialized.
C. Error: incorrect assignment statements.

TO-DO

- Start zyBooks chapter 8 and 9 exercises.
- Complete Project 4.

NOTE:

We’ll cover Recursion (chapter 8.9 to 8.14) and ArrayLists (9.9-9.13) after the Spring break.

Enjoy your Spring break!