L25

Thursday, January 27, 2022 12:56 AM

Different Methods

Q1. A method and its structure
In Java, a method (also called as function) is a named block of code that performs a task.

Like a mathematical function f(x) (read as ***f of x***), in Java, a method (or a function) can act on some input data to produce an output.

The input is not always compulsory, for example we could have a method called void printCurrentTime(), which could internally fetch the System's current time and print the same.

Similarly, the output of a function need not always be printed to a console, the function could store the output to a file, or a database or even return the calculated value to be used for further processing.

There are 5 basic parts to a method:

1. Method Name - the name used identify and invoke/execute the method.
2. Method Body - contains the code statements which will be executed when the method is called/invoked.
3. Method Parameters - [***optional***] contain the input values which are used by the statements in the method body.
4. Method Return Type - indicates what **type** of value is returned, if the method returns nothing, then void is used as the return type.
5. Modifiers - [***optional***] modifiers constitute one or more keywords that inform how the method can be used. We will learn more about them later.

Below is an example of a method which has parameters and returns a value:

```
public int sum(int num1, int num2) {
        return num1 + num2;
}
```

1. sum is the method name.
2. the content between opening { and closing } braces constitutes the method body.
3. int num1, int num2, which are passed between opening (and closing) parentheses are called the method parameters.
4. int which is before the method name sum is the return type of the method.
5. the keyword public used before the int return type is part of the method modifiers.

Below is an example of a method which **has no parameters** and **does not** return a value:

```
public void printHello() {
        System.out.println("Hello");
}
```

1. printHello is the method name.
2. the content between opening { and closing } braces constitutes the method body.
3. since there are no parameters we have empty opening (and closing) parentheses.
4. since the method does not return any value, we have provided void as the return type for the method.
5. the keyword public used before the void return type is part of the method modifiers.

- [ ] A method can be called only once.
- [ ] A method should contain at least one parameter.
- [ ] There should not be a space between method name and the opening parenthesis `(` which follows it.
- [ ] There should be only one space between the closing parenthesis `)` and the opening brace `{` which marks the beginning of the method body.
- [x] Method name should always start a lowercase letter.

Q2. A class with fields, constructor and toString() method

toString() is a special method used to convert an object's state information into a human readable text.

Note the spelling of toString()

The toString() method does not take any parameters and will always return a String.

See the code and retype the below code.

```java
package q11126;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q3. The toString() is a special method because when the object is involved in the string concatenation then the toString() method returns the String representation of that object. Hence a toString() method's return type is String and it does not take any parameters.

Identify and correct the errors based on the above information provided. Hint: Press Submit to find out the error.

```java
package q11125;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;
    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getId() {
        return id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public int getAge() {
        return age;
    }
    public void setGender(char gender) {
        this.gender = gender;
    }
    public char getGender() {
        return gender;
    }
    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q4. The toString() method like the constructor of a class is part of the class body. It must be written inside a class scope (inside the opening and closing brace of the class).

Hint: You can select some code and press ALT + Up key or ALT + Down key to move the selected code up and down. (**Note**: This works only in Windows and not in MAC and Linux)

Identify and correct the error.

q11128/Student.java | q11128/StudentMain.java

```java
package q11128;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String name, String id, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }


    public String toString() {
        return "Student [name = " + name + ", id = " + id + ", age = " + age + ", gender = " + gender + "]";
    }


}
```

Q5. Identify the errors and correct them.

q11129/Student.java | q11129/StudentMain.java

```java
package q11129;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public String toString() {
        return "Student [name = " + name + ", id = " + id + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q6. Whenever on compilation, you see an error saying: xyz cannot be resolved, it means that the compiler is unable to resolve the symbol xyz to a type.

On such errors first verify if the symbol xyz is spelt correctly.

Identify the error and correct it.

q11130/Student.java | q11130/StudentMain.java

```java
package q11130;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public String toString() {
        return "Student [name = " + name + ", id = " + id + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q7. You will notice that the line public class Student { is indented to the left most column.

At the same time you will also notice that the lines inside the class body are indented by one level.

Indenting code is indented to make code easily readable. It helps clearly identify the start and end of independent code blocks, like classes, methods loops etc.

One can either place the opening brace ({) in a new line or at the end of the statement, In all our examples we will place them at the end of the statement so that we can see more lines on the screen.

The general rules for indentation are as follows:

1. Press the tab key to indent and not space bar.
2. Indent the line which immediately follows the opening brace.
3. Indent the closing brace (}) to match with the beginning of the line which has the opening brace ({).

Identify the error and correct it.

Hint: Press the Submit button to find the error. Note if the correct operator is used for String concatenation.

q11131/Student.java

```java
package q11131;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public String toString() {
        return "Student [name = " + name + ", id = " + id + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

q11131/StudentMain.java

Q8. What are accessor methods?

The state information of an instance of Student class is stored in its instance fields.

These fields are usually marked as private.

These fields are initialized using the constructor. Hence, after the values are initialized, getXXX and setXXX methods are written to retrieve and modify their values.

A setXXX method is called a setter method. For a field named age, the setter method would be written as:

```
public void setAge(int age) {
        this.age = age;
}
```

A getXXX method is called a getter method. For a field named age, the getter method would be written as:

```
public int getAge() {
        return age;
}
```

These getter and setter methods are also called as accessor methods.

Note how the getter and setter method names follow camel-case naming convention.

See and retype the below code.

```java
package q11132;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q9. Identify the errors and correct them.

q11134/Student.java | q11134/StudentMain.java

```java
package q11134;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q10. A setter method will always have void as return type and takes the field's type as an argument into the method.

void return type indicates that the method does not return a value. It only performs some operation.

For example :

```
public void setName(String name) {
        this.name = name;
}
```

Identify the errors and correct them.

q11135/Student.java | q11135/StudentMain.java

```java
package q11135;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q11. A getter method of a field will always have the field's type as the return type and will not have any argument.

It does not take any argument since it simply returns the value of the field held by the instance.

For example :

```
public String getName() {
        return name;
}
```

Using the above information identify the errors in the below code and correct them. Press Submit to spot the errors.

q11136/Student.java | q11136/StudentMain.java

```java
package q11136;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q12. In the below you will find that some of the getter and setter methods have wrong return types.

Use the declared type of the field to correct the return types of getter methods and ponder upon what should be the return types of the setter methods to correct them.

Identify the errors and correct them. Hint: Press Submit to see the errors.

q11137/Student.java | q11137/StudentMain.java

```java
package q11137;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q13. In the below code you will find that wrong parameter types have been provided to the setter methods.

Use the declared field's type to correct the parameter type of the setter methods.

Identify the errors and correct them. Hint: Press Submit to view the errors.

q11138/Student.java | q11138/StudentMain.java

```java
package q11138;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q14. The toString() is a special method which is used to convert the state information stored in the fields into a String and return it.

Hence a toString() method's return type is String and it does not take any parameters.

Identify and correct the errors based on the above information provided. Hint: Press Submit to find out the error.

q11140/Student.java | q11140/StudentMain.java

```java
package q11140;
public class Student {
    private String id;
    private String name;
    private int age;
    private char gender;

    public Student(String id, String name, int age, char gender) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public char getGender() {
        return gender;
    }

    public String toString() {
        return "Student [id = " + id + ", name = " + name + ", age = " + age + ", gender = " + gender + "]";
    }
}
```

Q15. Difference between Parameters and Arguments

The list of variables/references that are present in the method declaration are called **parameters**.

When the method is invoked the actual values passed are called **arguments**.

Select all the correct statements for the below code:

```
public int sum(int num1, int num2) { // statement 1return num1 + num2;
}
int total = sum(2, 3); // statement 2
```

- [x] In statement 1, `num1` and `num2` are called parameters.
- [ ] In statement 1, `num1` and `num2` are called arguments.
- [x] In statement 2, values `2` and `3` are called arguments.
- [ ] In statement 2, values `2` and `3` are called parameters.