Solutions to Quick Check Questions

8

Exceptions and Assertions

8.1 Catching Exceptions

1. What will be displayed on the console window when the following code is executed and the user enters abc123 and 14?

```
Scanner scanner = new Scanner(System.in);
try {
    int num1 = scanner.nextInt();
    System.out.println("Input 1 accepted");
    int num2 = scanner.nextInt();
    System.out.println("Input 2 accepted");
} catch (InputMismatchException e) {
    System.out.println("Invalid Entry");
}
```

Answer:

```
Invalid Entry
```

2. What is wrong with the following code? It attempts to loop until the valid input is entered.

```
Scanner scanner = new Scanner(System.in);
try {
    while (true) {
        System.out.println("Ener input:");
        int num  = scanner.nextInt();
    }
} catch (InputMismatchException e) {
    scanner.next()
    System.out.println("Invalid Entry");
}
```

The try-catch block must be inside the loop. As the code is written now, when there's an exception, the control jumps out of the loop, so the enduser won't get another try.

8.2 Throwing Exceptions and Multiple catch Blocks

1. What's wrong with the following code? Identify all errors.

```
        Scanner scanner = new Scanner(System.in);
        try {
            int num = scanner.nextInt();
            if (num > 100) {
A  ------->     catch new Exception("Out of bound");
            }
        } catch (InputMismatchException e) {
            System.out.println("Invalid Entry");
B  ---> } finally(Exception e) {
            System.out.println("DONE");
        }
```

A. The correct reserved word here is throw.

B. The finally *clause does not include a parameter.*

2. Determine the output of the following code when the input **a12** is entered.

```
Scanner scanner = new Scanner(System.in);
try {
    int num = scanner.nextInt();
    if (num < 0) {
        throw new Exception("No negative");
    }
} catch (InputMismatchException e) {
    System.out.println("Invalid Entry");
} catch (Exception e) {
    System.out.println("Error: "+ e.getMessage());
} finally {
    System.out.println("DONE");
}
```

Answer:

```
Invalid Entry
Done
```

3. Determine the output of the following code when the input **a12** is entered.

```
Scanner scanner = new Scanner(System.in);
try {
    int num = scanner.nextInt();
    if (num < 0) {
        throw new Exception("No negative");
    }
} catch (Exception e) {
    System.out.println("Error: "+ e.getMessage());
} catch (InputMismatchException e) {
    System.out.println("Invalid Entry");
}
```

Answer:

```
Error: No negative
```

8.3 Propagating Exceptions

1. What's wrong with the following code?

```
public void check(int num) {
    if (num < 0) {
        throw new Exception();
    }
}
```

The method header must include the clause throws Exception.

2. What is the difference between the reserved words throw and throws?

The reserved word throw *is used when you throw an exception. The reserved word* throws *is used in the method header if this method includes a statement that can throw an exception.*

3. What's wrong with the following code?

```
public InputMismatchException getData( ) {

    Scanner scanner = new Scanner(System.in);
    try {
        System.out.println("Input: ");
        int num = scanner.nextInt();
        return num;
    }
}
```

The exception type such as InputMismatchException cannot be a return type. The correct declaration is

```
public int getData() throws NumberFormatException {
...
}
```

The try block must have at least one matching catch block.

8.4 Types of Exceptions

1. Is this code wrong?

```
public void check(int num) {
    if (num < 0) {
        throw new IllegalArgumentException();
    }
}
```

   The code is correct. Runtime exceptions do not have to be declared in the method header.

2. What is the difference between the checked and unchecked exceptions?

   The checked expections are those checked at the compile time. The unchecked exceptions are those detected at runtime. For this reason, unchecked exceptions are also called runtime exceptions.

8.5 Programmer-Defined Exceptions

1. When do we want to define a customized exception class?

   When we need to include additional information to a thrown exception.

2. Should a customized exception class be a checked or unchecked exception?

   Logically, a customized exception class should be a checked exception because we want it to be checked at the compile time.

8.6 Assertions

1. Why is the following code wrong?

```
public void doWork(int num) {
    assert num > 0;
    total += num;
}
```

From a design standpoint, assertions are not recommended for checking the validity of arguments. Use assertions for detecting internal errors. Use exceptions to detect the misuse of a method by the client programmers.

2. Name three types of assertions?

precondition assertion
postcondition assertion
control-flow invariant

8.7 Sample Development: Keyless Entry System

No Quick Check Questions.