Java Programming 5-2: Input and Output

Fundamentals Practice Activities

Vocabulary Definitions

1. **Serialization**: The process of converting a stream of data into an object.
2. **Stream**: Sequences of bytes or characters transmitted from one program to another program or a file.
3. **Escape Character**: A backslash in front of another character makes Java treat the next character as an ordinary character.
4. **Deserialization**: The process of converting a stream of data into an object.
5. **Input Stream**: Sequences of bytes or characters transmitted from another program or user's activity.
6. **Symbolic Link**: A file name that maps to another file.
7. **Absolute Path**: A path that is direct rather than indirect.
8. **Error Output Stream**: This is the output stream for error raised by a program, also known as the second channel.
9. **Path**: The join of two related paths, or the remaining part of a join after subtracting part of the path.
10. **Standard Streams**: Types of standard input, output, and error; and qualified object types in Java.
11. **Standard Output Stream**: The output for debugging messages and ordinary reports and messages.
12. **Input Device**: Any keyboard, mouse, or touchscreen input to a program.
13. **Serialization**: The process of converting an object or file to a series of bytes.

Try It/Solve It

1. Resolve and Print a Path

Here's a Java class that demonstrates resolving and printing a `Path`:

java

```
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
public class PathTest {
   public static void main(String[] args) {
      // Create an instance of the Path interface
      Path path = Paths.get("C:/JavaProgramming/employees.txt");

      // Print the constructed Path
      System.out.println("Path: " + path);
   }
}
```

2. Serialization and Deserialization

Here's how to handle serialization and deserialization in Java. We'll start by modifying the `Employee` class to implement `Serializable`, and then we'll create methods to serialize and deserialize an `Employee` object.

`Employee` Class (Serialization)

Ensure your `Employee` class implements `Serializable`:

```
import java.io.Serializable;

public class Employee implements Serializable {
   private static final long serialVersionUID = 1L; // A unique ID for serialization
   private String name;
   private String username;
   private String email;
   private String password;

   public Employee(String name, String username, String email, String password) {
      this.name = name;
      this.username = username;
      this.email = email;
      this.password = password;
   }

   // Getters and setters for the fields
   // toString() method for easy display
   @Override
   public String toString() {
      return "Employee Details\n" +
           "Name: " + name + "\n" +
           "Username: " + username + "\n" +
           "Email: " + email + "\n" +
           "Initial Password: " + password;
   }
}
```

`AccountGenerator` Class (Serialization and Deserialization)

Here's how to serialize and deserialize an `Employee` object:

```
import java.io.*;


public class AccountGenerator {


   public static void serializeData(Employee emp) {

      try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("employee.ser"))) {

         out.writeObject(emp);

         System.out.println("Employee serialized to employee.ser");

      } catch (IOException e) {

         System.err.println("Error serializing Employee: " + e.getMessage());

      }

   }


   public static Employee deserializeData() throws ClassNotFoundException {

      try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream("employee.ser"))) {

         return (Employee) in.readObject();

      } catch (IOException e) {

         System.err.println("Error deserializing Employee: " + e.getMessage());

         return null;

      }

   }
```

```
    public static void main(String[] args) {

        // Create an Employee object

        Employee emp = new Employee("June Summers", "june.summers",
"jsummers@oracleacademy.Test", "J*l**.s*");



        // Serialize the Employee object

        serializeData(emp);



        // Deserialize the Employee object

        try {

            Employee deserializedEmp = deserializeData();

            if (deserializedEmp != null) {

                System.out.println("Deserialized Employee:");

                System.out.println(deserializedEmp);

            }

        } catch (ClassNotFoundException e) {

            System.err.println("Class not found during deserialization: " + e.getMessage());

        }

    }

}
```