CS F213 - Object Oriented Programming

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 P, NAB
Consultation: Appointment by e-mail

https://github.com/JenniferRanjani/Object-Oriented-Programming-with-Java

BITS Pilani
Pilani Campus

User Defined Exception

- Helps the user to create their own exception for situations specific to the applications.
- The user defined exception class is derived from the Exception class
- Exception class does not have methods on its own. But it inherits methods provided by the Throwable class.
- Overriding the toString() method helps to provide a description about the exception.

Chained Exceptions

- Allows to associate another exception with an exception.
- The second exception describes the cause of the first exception.
- Chained exception methods supported by Throwable are getCause() and initCause()

Exception - Example

```java
class test
{
public static void main(String args[]) {
try{
NumberFormatException e =new NumberFormatException("Outer");
e.initCause(new ArithmeticException("Inner"));
throw e;
}
catch(NumberFormatException e) {
System.out.println(e);
System.out.println(e.getCause());
}
}
}
```

Additional Exception Features

- try-with resources
- multi-catch

Automatic resource management

- Traditionally, an explicit call to the close() method should be called when the resources is no longer required.

- Automatic resource management introduced in JDK 7 uses an expanded version of try statement.

- It prevents memory leaks.

try-with resource - example

```
class test
{
private final static String FILENAME = "file1.txt";
    public static void main(String[] args) {
        try(BufferedReader rd = new BufferedReader(new FileReader(FILENAME))) {
            String inputLine = null;

            while((inputLine = rd.readLine()) != null)
                System.out.println(inputLine);
        }
        catch (IOException ex) {
            System.err.println("An IOException was caught: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

multi-catch Example

```
class test
{
public static void main(String args[]) {
int a = 24, b = 0, c;
int d[] = {1,2,3,4};
try{
//c = a/b;
d[12]=9;
}
catch(ArithmeticException | ArrayIndexOutOfBoundsException e) {
System.out.println(e);
}
}
}
```

Serialization

- Process of writing the state of an object to a byte stream which can be restored using deserialization.
- It is useful in remote method invocation (RMI), which allows Java object on one machine to invoke a method of a Java object on a different machine.
- Java object is supplied as a argument to that method
- A directed graph is used to represent the objects and its relationship, when it has reference to other objects, which in turn have references to objects.
- When the object at the top of the object graph is serialized all the other objects are recursively located and serialized.

Required Interfaces

- Serializable: It has no members, it is used to indicate that a class may be serialized. If a class is serialized, all of its sub classes are also serializable.

- Externalizable: When the programmer need to have control over serialization of objects
  - Eg.: use of compression or encryption technique
  - void readExternal(ObjectInput instream) throws IOException, ClassNotFoundException
  - void writeExternal(ObjectInput instream) throws IOException

Required Interfaces

- ObjectOutput: extends the DataOutput and AutoCloseable interfaces
  - close, flush, write, writeObject

- ObjectInput: extends the DataInput and AutoCloseable interfaces
  - available, close, read, readObject, skip

Required Classes

- ObjectOutputStream: extends the OutputStream class and implements the ObjectOutput interfaces. It is responsible for writing objects to a stream.
  - Write(Byte,Boolean,Char,Chars,Double,Float,Int,Short,Long,Object)

- ObjectInputStream:extends the InputStream class and implements the ObjectInput interface.

Transient Modifier

- When an instance variable is declared as transient, the value of the instance variable need not persist when the object is stored.
- The object would be written in the storage area but the contents of the transient variable would not be saved.
- Transient has no impact on static and final variables

Review Questions

- Can we create checked and unchecked user defined exceptions? If so, For the Account example, create checked as well as unchecked exceptions for the name field. If the name field is empty raise a MissingNameException.

- Try serializing the name and the amount fields of the Account class.