Assignment 1: Arithmetic Exception

```
public class ArithmeticExceptionDemo {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
            System.out.println(a);
        } catch (ArithmeticException e) {
            System.out.println("Division by zero is not allowed");
        }
    }
}

Output:
Division by zero is not allowed
```

Assignment 2: Array Index Exception

```
public class ArrayIndexExceptionDemo {
    public static void main(String[] args) {
        try {
            int[] arr = {1,2,3};
            System.out.println(arr[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Invalid array index");
        }
    }
}

Output:
Invalid array index
```

Assignment 3: Null Pointer Exception

```
public class NullPointerExceptionDemo {
    public static void main(String[] args) {
        try {
            String s = null;
            System.out.println(s.length());
        } catch (NullPointerException e) {
            System.out.println("Null pointer exception handled");
        }
    }
}

Output:
Null pointer exception handled
```

Assignment 4: Multiple Catch

```
public class MultipleCatchDemo {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic exception");
        } catch (Exception e) {
            System.out.println("General exception");
        }
    }
}

Output:
Arithmetic exception
```

Assignment 5: Finally Block

```
public class FinallyBlockDemo {
    public static void main(String[] args) {
        try {
            int x = 10 / 2;
            System.out.println(x);
        } finally {
            System.out.println("Finally block executed");
        }
    }
}

Output:
5
Finally block executed
```

Assignment 6: Nested Try Catch

```
public class NestedTryCatchDemo {
    public static void main(String[] args) {
        try {
            try {
                int a = 10 / 0;
            } catch (ArithmeticException e) {
                System.out.println("Inner catch");
            }
            int[] arr = new int[3];
            arr[5] = 10;
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Outer catch");
        }
    }
}

Output:
Inner catch
Outer catch
```

Assignment 7: User Input Validation

```
public class UserInputValidationDemo {
    public static void main(String[] args) {
        int age = 15;
        try {
            if(age < 18) {
                throw new Exception("Not eligible to vote");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

Output:
Not eligible to vote
```

Assignment 8: Custom Exception

```
class InvalidBalanceException extends Exception {
    public InvalidBalanceException(String msg) {
        super(msg);
    }
}

public class CustomExceptionDemo {
```

```
    public static void main(String[] args) {
        int balance = 500;
        try {
            if(balance < 1000) {
                throw new InvalidBalanceException("Insufficient balance");
            }
        } catch (InvalidBalanceException e) {
            System.out.println(e.getMessage());
        }
    }
}

Output:
Insufficient balance
```

Assignment 9: Exception Propagation

```
public class ExceptionPropagationDemo {
    static void m() {
        int a = 10 / 0;
    }
    static void n() {
        m();
    }
    public static void main(String[] args) {
        try {
            n();
        } catch (ArithmeticException e) {
            System.out.println("Exception handled in main");
        }
    }
}

Output:
Exception handled in main
```

Assignment 10: Rethrow Exception

```
public class RethrowExceptionDemo {
    public static void main(String[] args) {
        try {
            try {
                int a = 10 / 0;
            } catch (ArithmeticException e) {
                System.out.println("Caught exception");
                throw e;
            }
        } catch (ArithmeticException e) {
            System.out.println("Rethrown exception handled");
        }
    }
}

Output:
Caught exception
Rethrown exception handled
```

Assignment 11: Exception in Method Overriding

```
class ParentClass {
    void show() throws ArithmeticException {
        System.out.println("Parent method");
    }
}

public class ChildClass extends ParentClass {
    void show() {
        System.out.println("Child method");
```

```
    }
    public static void main(String[] args) {
        ChildClass obj = new ChildClass();
        obj.show();
    }
}

Output:
Child method
```

Assignment 12: Multiple Custom Exceptions

```
class InvalidAgeException extends Exception {
    InvalidAgeException(String msg) {
        super(msg);
    }
}

public class MultipleCustomExceptionDemo {
    public static void main(String[] args) {
        int age = 16;
        try {
            if(age < 18) {
                throw new InvalidAgeException("Age not valid");
            }
        } catch (InvalidAgeException e) {
            System.out.println(e.getMessage());
        }
    }
}

Output:
Age not valid
```