# L49

Friday, February 4, 2022 9:14 PM

# Usage of throw, throws and Writing User defined Exceptions

## Usage of throw, throws

Q1. The **throw** keyword is used to write the throw statement which causes the exception to be thrown.

The syntax for the **throw statement** is as below:
**throw***aThrowableInstance*;
For example:
**throw** new Exception("The world is about to end!");
As you can notice the throw clause should be followed by an instance of Throwable or one of its subclasses.

We can also throw custom exceptions, about which we will learn later.

The throws keyword is used in the method or constructor declaration. It is used to inform (or list) all the **checked** exceptions which the method or constructor body can throw during execution.

Note that the throws clause in the method or constructor declaration **need not** list the **unchecked** exceptions that are thrown by the code in the method or constructor body. The syntax for the throws is as given below:
*methodModifiersListreturnType* methodName(*parameterList*) **throws*ExceptionClassName1, ExceptionClassName2, ...*** {
}
For example:

```
public void setAge(int age) throwsInvalidAgeException {
        if (age < 0 || age > 999) { //assuming super-humans can live 999 yearsthrow new InvalidAgeException("Invalid age. Valid range for age is between 0 and 999.");
        }
        this.age = age;
}
```

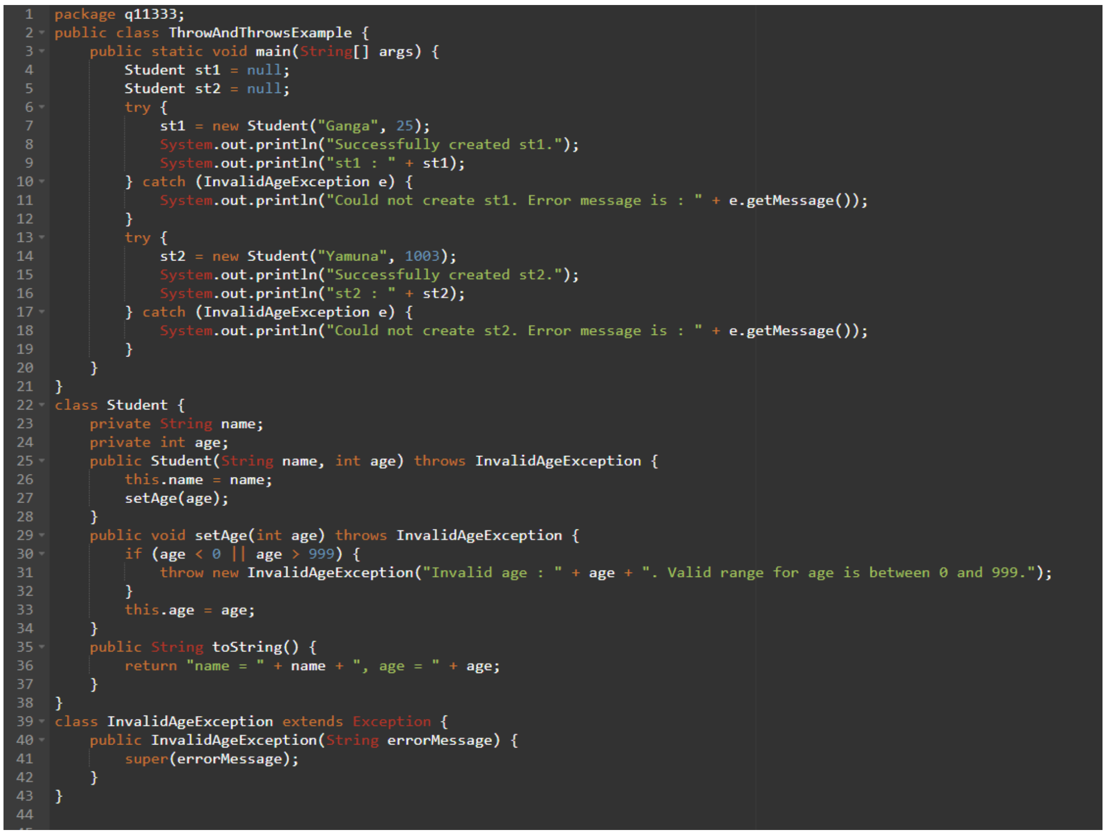
The throw clause can throw **only a single exception** at a time.

However, the throws clause **can specify multiple exceptions** the method or constructor throws.

Note that a method or a constructor which does not want to handle a checked exception can let it go out of it, by including that exception class name in its throws declaration.

For example, in the below code the Student(String name, int age) constructor does not handle the InvalidAgeException which is thrown by the setAge method using a try-catch block. Instead the constructor included InvalidAgeException in its throws clause.

See and retype the below code.



```java
package q11333;
public class ThrowAndThrowsExample {
    public static void main(String[] args) {
        Student st1 = null;
        Student st2 = null;
        try {
            st1 = new Student("Ganga", 25);
            System.out.println("Successfully created st1.");
            System.out.println("st1 : " + st1);
        } catch (InvalidAgeException e) {
            System.out.println("Could not create st1. Error message is : " + e.getMessage());
        }
        try {
            st2 = new Student("Yamuna", 1003);
            System.out.println("Successfully created st2.");
            System.out.println("st2 : " + st2);
        } catch (InvalidAgeException e) {
            System.out.println("Could not create st2. Error message is : " + e.getMessage());
        }
    }
}
class Student {
    private String name;
    private int age;
    public Student(String name, int age) throws InvalidAgeException {
        this.name = name;
        setAge(age);
    }
    public void setAge(int age) throws InvalidAgeException {
        if (age < 0 || age > 999) {
            throw new InvalidAgeException("Invalid age : " + age + ". Valid range for age is between 0 and 999.");
        }
        this.age = age;
    }
    public String toString() {
        return "name = " + name + ", age = " + age;
    }
}
class InvalidAgeException extends Exception {
    public InvalidAgeException(String errorMessage) {
        super(errorMessage);
    }
}
```

Q2. Write a Java program for creation of illustrating **throw**.

Write a class ThrowExample contains a method **checkEligibilty(int age, int weight)** which throws an **ArithmeticException** with a message "**Student is not eligible for registration**" when **age < 12** and **weight < 40**, otherwise it prints "**Student Entry is Valid!!**".

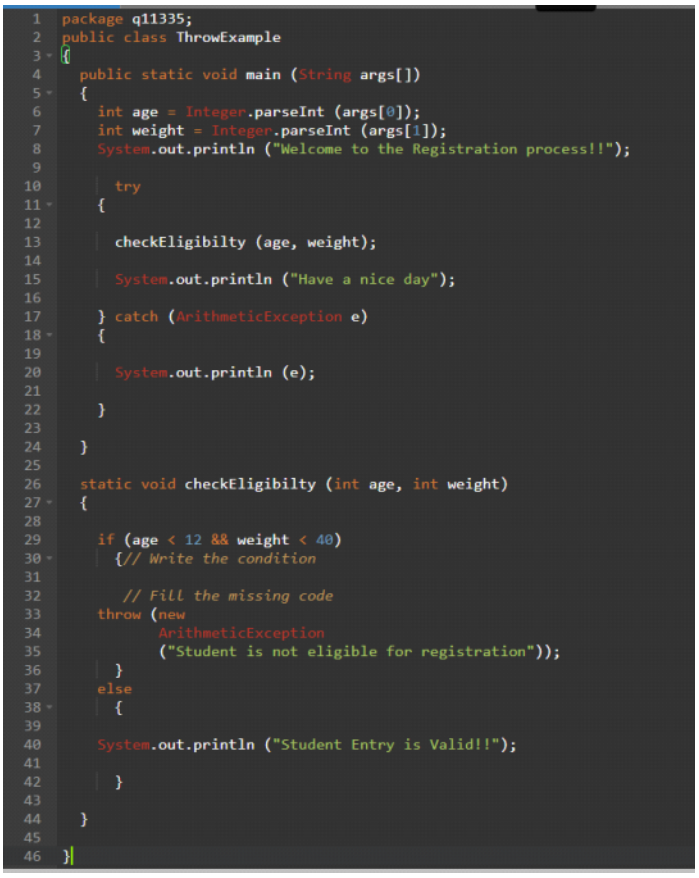
Write the **main()** method in the same class which will receive two arguments as **age** and **weight**, convert them into **integers**.

For example, if the given data is 9 and 35 then the output should be:

```
Welcome to the Registration process!!
java.lang.ArithmeticException: Student is not eligible for registration
```

For example, if the given data is 15 and 41 then the output should be:

```
Welcome to the Registration process!!
Student Entry is Valid!!
Have a nice day
```



```java
package q11335;
public class ThrowExample
{
  public static void main (String args[])
  {
    int age = Integer.parseInt (args[0]);
    int weight = Integer.parseInt (args[1]);
    System.out.println ("Welcome to the Registration process!!");

      try
    {

      checkEligibilty (age, weight);

      System.out.println ("Have a nice day");

    } catch (ArithmeticException e)
    {

      System.out.println (e);

    }

  }

  static void checkEligibilty (int age, int weight)
  {

    if (age < 12 && weight < 40)
      {// Write the condition

       // Fill the missing code
    throw (new
           ArithmeticException
           ("Student is not eligible for registration"));
      }
    else
      {

    System.out.println ("Student Entry is Valid!!");

      }

  }

}
```

# Writing User defined Exceptions

Q1. It is very easy to write a custom exception class. All that we have to do is write a class and extend the Exception class.

Even though Throwable is the super class of all the exception and error classes, we normally extend the Exception class and not Throwable.

The simple rule for naming a custom exception class is as below:

<***ErrorCondition***>Exception

Some examples for custom exception class names are given below:

InvalidAgeException
InvalidNameException
InvalidEmailException

Note that it is a good practice to always end the name of the exception class with Exception, for easy identification.

Below is an example of a custom exception class:

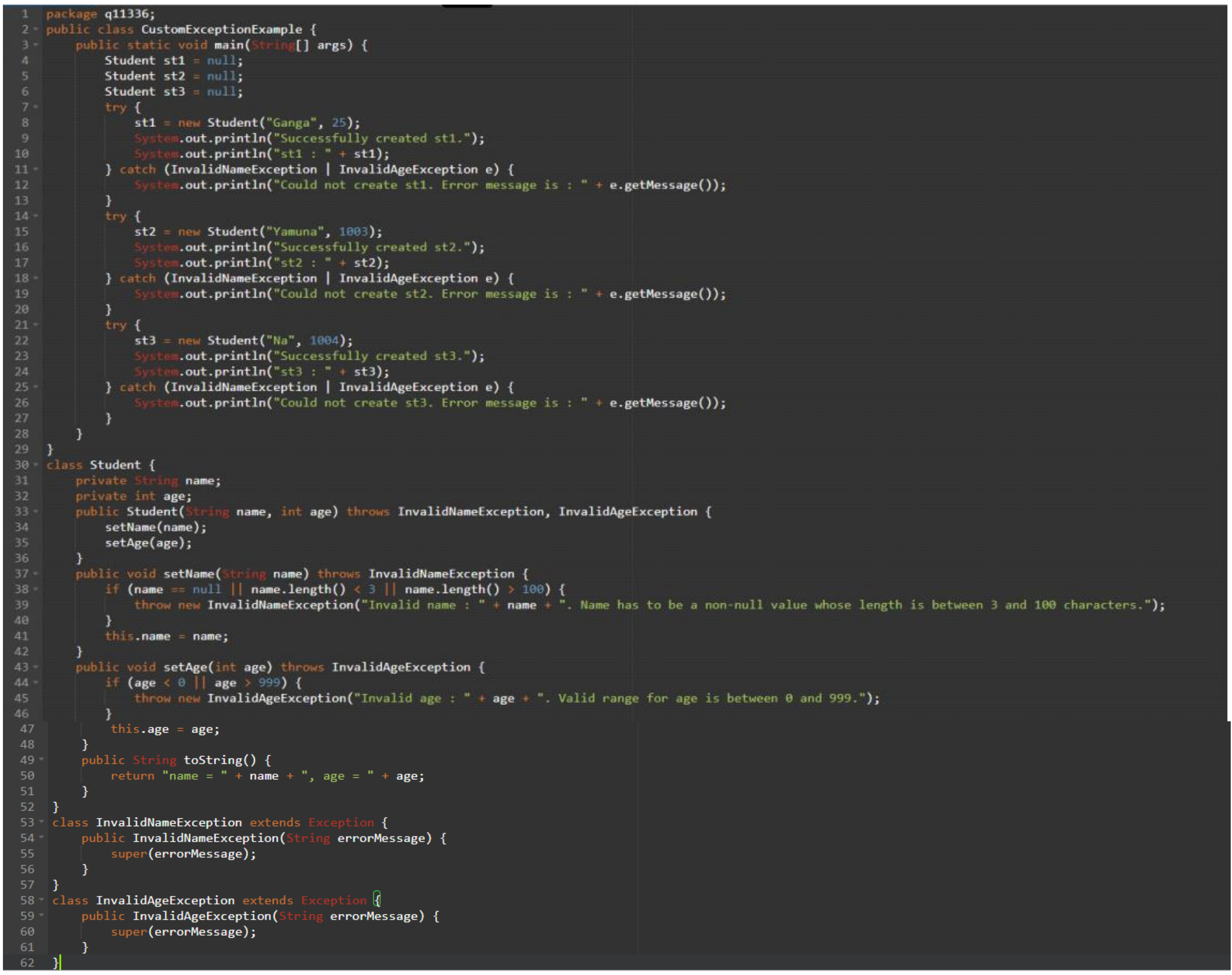
```java
class InvalidAgeException extendsException {
        public InvalidAgeException(String errorMessage) {
                super(errorMessage);
        }
}
```

As you can notice in the above code, all we need to do to write a custom exception class is:

1. Write a class name which ends with **Exception**.
2. Extend the Exception class using the extends clause in the class declaration statement.
3. And write a constructor which accepts an error message as a Sting.
4. In the constructor call the constructor in the super class and pass the error message, using **super**(errorMessage) call.

See and retype the below code.

After executing the below code you will notice that, while Student constructor is called during the creation of st3 (in line no: 21), the call to setAge(age); (in line no: 34) is skipped because the previous statement setName(name); (at line no: 33) will throw a InvalidNameException and the control abruptly is transferred out of the constructor.



```java
package q11336;
public class CustomExceptionExample {
    public static void main(String[] args) {
        Student st1 = null;
        Student st2 = null;
        Student st3 = null;
        try {
            st1 = new Student("Ganga", 25);
            System.out.println("Successfully created st1.");
            System.out.println("st1 : " + st1);
        } catch (InvalidNameException | InvalidAgeException e) {
            System.out.println("Could not create st1. Error message is : " + e.getMessage());
        }
        try {
            st2 = new Student("Yamuna", 1003);
            System.out.println("Successfully created st2.");
            System.out.println("st2 : " + st2);
        } catch (InvalidNameException | InvalidAgeException e) {
            System.out.println("Could not create st2. Error message is : " + e.getMessage());
        }
        try {
            st3 = new Student("Na", 1004);
            System.out.println("Successfully created st3.");
            System.out.println("st3 : " + st3);
        } catch (InvalidNameException | InvalidAgeException e) {
            System.out.println("Could not create st3. Error message is : " + e.getMessage());
        }
    }
}
class Student {
    private String name;
    private int age;
    public Student(String name, int age) throws InvalidNameException, InvalidAgeException {
        setName(name);
        setAge(age);
    }
    public void setName(String name) throws InvalidNameException {
        if (name == null || name.length() < 3 || name.length() > 100) {
            throw new InvalidNameException("Invalid name : " + name + ". Name has to be a non-null value whose length is between 3 and 100 characters.");
        }
        this.name = name;
    }
    public void setAge(int age) throws InvalidAgeException {
        if (age < 0 || age > 999) {
            throw new InvalidAgeException("Invalid age : " + age + ". Valid range for age is between 0 and 999.");
        }
        this.age = age;
    }
    public String toString() {
        return "name = " + name + ", age = " + age;
    }
}
class InvalidNameException extends Exception {
    public InvalidNameException(String errorMessage) {
        super(errorMessage);
    }
}
class InvalidAgeException extends Exception {
    public InvalidAgeException(String errorMessage) {
        super(errorMessage);
    }
}
```

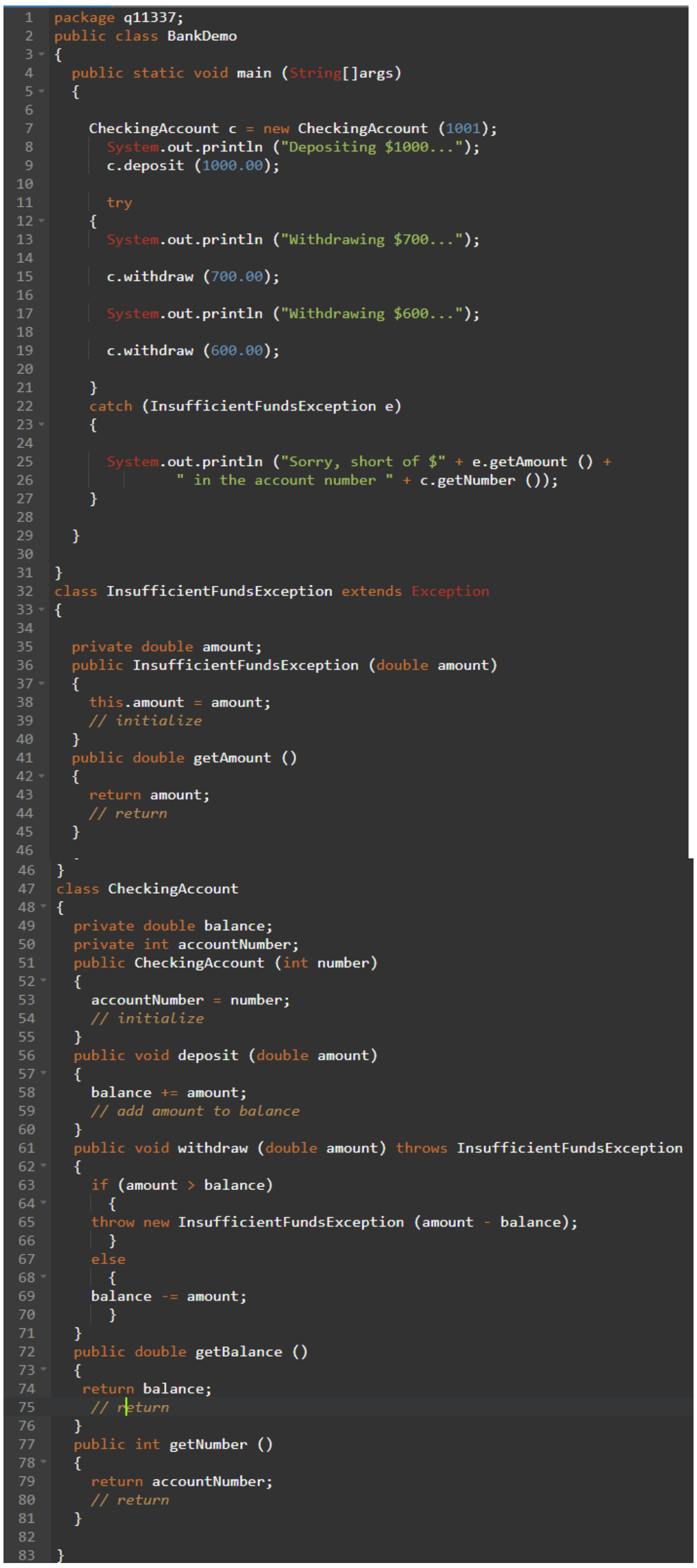
Q2. Write a Java program to illustrate **user-defined exceptions.**

Write the class InsufficientFundsException with

- private double member **amount**
- a parameterized **constructor** to initialize the amount
- a method **getAmount()** to return amount.

Write another class CheckingAccount with

- two private members **balance** and **accountNumber**
- a parameterized **constructor** to initialize the accountNumber
- method **deposit()** to add amount to the balance
- method **withdraw()** to debit amount from balance if sufficient balance is available, otherwise throw an exception **InsufficientFundsException()** with how much amount needed extra
- method **getBalance()** to return balance.
- method **getNumber()** to return accountNumber.



```java
package q11337;
public class BankDemo
{
  public static void main (String[]args)
  {

    CheckingAccount c = new CheckingAccount (1001);
      System.out.println ("Depositing $1000...");
      c.deposit (1000.00);

      try
    {
      System.out.println ("Withdrawing $700...");

      c.withdraw (700.00);

      System.out.println ("Withdrawing $600...");

      c.withdraw (600.00);

    }
    catch (InsufficientFundsException e)
    {

      System.out.println ("Sorry, short of $" + e.getAmount () +
              " in the account number " + c.getNumber ());
    }

  }

}
class InsufficientFundsException extends Exception
{

  private double amount;
  public InsufficientFundsException (double amount)
  {
    this.amount = amount;
    // initialize
  }
  public double getAmount ()
  {
    return amount;
    // return
  }

}
class CheckingAccount
{
  private double balance;
  private int accountNumber;
  public CheckingAccount (int number)
  {
    accountNumber = number;
    // initialize
  }
  public void deposit (double amount)
  {
    balance += amount;
    // add amount to balance
  }
  public void withdraw (double amount) throws InsufficientFundsException
  {
    if (amount > balance)
      {
    throw new InsufficientFundsException (amount - balance);
      }
    else
      {
    balance -= amount;
      }
  }
  public double getBalance ()
  {
   return balance;
    // return
  }
  public int getNumber ()
  {
    return accountNumber;
    // return
  }

}
```

Q3. Write a Java program to illustrate the concept **creation of own exceptions**.

Write the class NumberRangeException which is inherited from Exception, contains only a default constructor which will print the message "**Please enter a number between 25 and 50**".

Write the class MyException with the **main()** method which will receive only one argument and convert that into **int**.

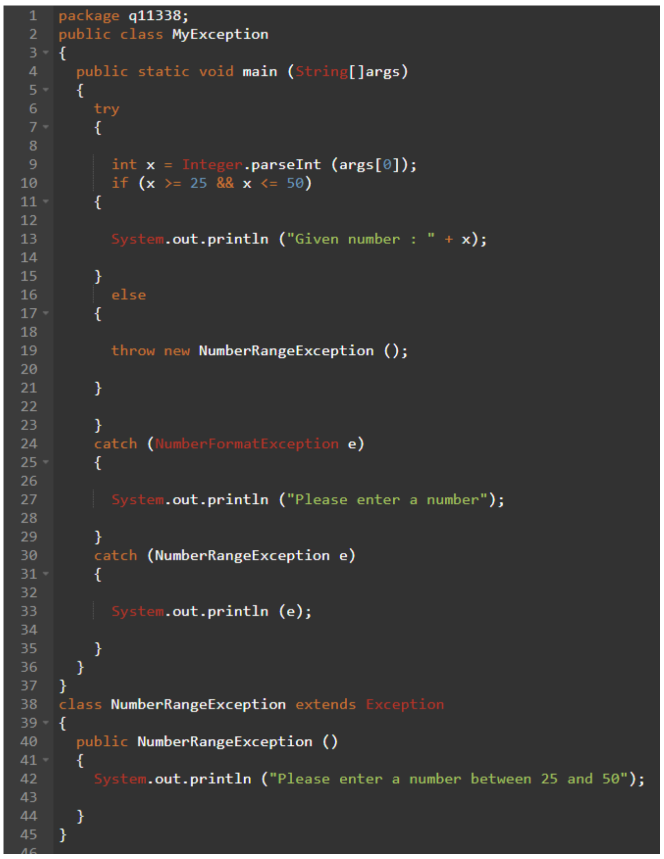
If the given integer is in between **25** and **50** print the given value, otherwise throw the **NumberRangeException()**.

For example, if the given integer is **27** then the output should be:

```
Given number : 27
```

For example, if the given integer is **62** then the output should be:

```
Please enter a number between 25 and 50
NumberRangeException
```



```java
package q11338;
public class MyException
{
  public static void main (String[]args)
  {
    try
    {

      int x = Integer.parseInt (args[0]);
      if (x >= 25 && x <= 50)
    {

      System.out.println ("Given number : " + x);

    }
      else
    {

      throw new NumberRangeException ();

    }

    }
    catch (NumberFormatException e)
    {

      System.out.println ("Please enter a number");

    }
    catch (NumberRangeException e)
    {

      System.out.println (e);

    }
  }
}
class NumberRangeException extends Exception
{
  public NumberRangeException ()
  {
    System.out.println ("Please enter a number between 25 and 50");

  }
}
```