1. Write product service class with a method to check product weight. If weight is less than 100 throw a custom Exception called : “Invalid product exception”.
   1. In the main class test with different weights and handle the exception by printing the stack trace.
   2. Try not to use intelij idea and use notepad++ instead.
2. Write a method to make stack over flow error.

Exam:

**1.** Which of the following statements are true? (Choose all that apply)

**A.** Runtime exceptions are the same thing as checked exceptions.

**B.** Runtime exceptions are the same thing as unchecked exceptions.

**C.** You can declare only checked exceptions.

**D.** You can declare only unchecked exceptions.

**E.** You can handle only Exception subclasses.

**2.** Which of the following pairs fill in the blanks to make this code compile? (Choose all that

apply)

7: public void ohNo() \_\_\_\_\_ Exception {

8: \_\_\_\_\_\_\_\_\_\_\_\_\_ Exception();

9: }

**A.** On line 7, fill in throw

**B.** On line 7, fill in throws

**C.** On line 8, fill in throw

**D.** On line 8, fill in throw new

**E.** On line 8, fill in throws

**F.** On line 8, fill in throws new

**3.** When are you required to use a finally block in a regular try statement (not a try-withresources)?

**A.** Never.

**B.** When the program code doesn’t terminate on its own.

**C.** When there are no catch blocks in a try statement.

**D.** When there is exactly one catch block in a try statement.

**E.** When there are two or more catch blocks in a try statement.

**4.** Which exception will the following throw?

Object obj = new Integer(3);

String str = (String) obj;

System.out.println(str);

**A.** ArrayIndexOutOfBoundsException

**B.** ClassCastException

**C.** IllegalArgumentException

**D.** NumberFormatException

**E.** None of the above.

**5.** Which of the following exceptions are thrown by the JVM? (Choose all that apply)

**A.** ArrayIndexOutOfBoundsException

**B.** ExceptionInInitializerError

**C.** java.io.IOException

**D.** NullPointerException

**E.** NumberFormatException

**6.** What will happen if you add the statement System.out.println(5 / 0); to a working

main() method?

**A.** It will not compile.

**B.** It will not run.

**C.** It will run and throw an ArithmeticException.

**D.** It will run and throw an IllegalArgumentException.

**E.** None of the above.

**7.** What is printed besides the stack trace caused by the NullPointerException from line 16?

1: public class DoSomething {

2: public void go() {

3: System.out.print("A");

4: try {

5: stop();

6: } catch (ArithmeticException e) {

7: System.out.print("B");

8: } finally {

9: System.out.print("C");

10: }

11: System.out.print("D");

12: }

13: public void stop() {

14: System.out.print("E");

15: Object x = null;

16: x.toString();

17: System.out.print("F");

18: }

19: public static void main(String[] args) {

20: new DoSomething().go();

21: }

22: }

**A.** AE

**B.** AEBCD

**C.** AEC

**D.** AECD

**E.** No output appears other than the stack trace.

**8.** What is the output of the following snippet, assuming a and b are both 0?

3: try {

4: return a / b;

5: } catch (RuntimeException e) {

6: return -1;

7: } catch (ArithmeticException e) {

8: return 0;

9: } finally {

10: System.out.print("done");

11: }

**A.** -1

**B.** 0

**C.** done-1

**D.** done0

**E.** The code does not compile.

**F.** An uncaught exception is thrown.

**9.** What is the output of the following program?

1: public class Laptop {

2: public void start() {

3: try {

4: System.out.print("Starting up ");

5: throw new Exception();

6: } catch (Exception e) {

7: System.out.print("Problem ");

8: System.exit(0);

9: } finally {

10: System.out.print("Shutting down ");

11: }

12: }

13: public static void main(String[] args) {

14: new Laptop().start();

15: } }

**A.** Starting up

**B.** Starting up Problem

**C.** Starting up Problem Shutting down

**D.** Starting up Shutting down

**E.** The code does not compile.

**F.** An uncaught exception is thrown.

**10.** What is the output of the following program?

1: public class Dog {

2: public String name;

3: public void parseName() {

4: System.out.print("1");

5: try {

6: System.out.print("2");

7: int x = Integer.parseInt(name);

8: System.out.print("3");

9: } catch (NumberFormatException e) {

10: System.out.print("4");

11: }

12: }

13: public static void main(String[] args) {

14: Dog leroy = new Dog();

15: leroy.name = "Leroy";

16: leroy.parseName();

17: System.out.print("5");

18: } }

**A.** 12

**B.** 1234

**C.** 1235

**D.** 124

**E.** 1245

**F.** The code does not compile.

**G.** An uncaught exception is thrown.

**11.** What is the output of the following program?

1: public class Cat {

2: public String name;

3: public void parseName() {

4: System.out.print("1");

5: try {

6: System.out.print("2");

7: int x = Integer.parseInt(name);

8: System.out.print("3");

9: } catch (NullPointerException e) {

10: System.out.print("4");

11: }

12: System.out.print("5");

13: }

14: public static void main(String[] args) {

15: Cat leo = new Cat();

16: leo.name = "Leo";

17: leo.parseName();

18: System.out.print("6");

19: }

20: }

**A.** 12, followed by a stack trace for a NumberFormatException

**B.** 124, followed by a stack trace for a NumberFormatException

**C.** 12456

**D.** 12456

**E.** 1256, followed by a stack trace for a NumberFormatException

**F.** The code does not compile.

**G.** An uncaught exception is thrown.

**12.** What is printed by the following? (Choose all that apply)

1: public class Mouse {

2: public String name;

3: public void run() {

4: System.out.print("1");

5: try {

6: System.out.print("2");

7: name.toString();

8: System.out.print("3");

9: } catch (NullPointerException e) {

10: System.out.print("4");

11: throw e;

12: }

13: System.out.print("5");

14: }

15: public static void main(String[] args) {

16: Mouse jerry = new Mouse();

17: jerry.run();

18: System.out.print("6");

19: } }

answer ==> 1 2 4