Excepciones en java

1. NullPointerException: Se produce cuando intentas acceder a un objeto que no ha sido inicializado (null). Ejemplo:

```
String str = null;
int length = str.length(); // Esto lanzará NullPointerException
```

2. ArrayIndexOutOfBoundsException: Se produce cuando intentas acceder a un índice inválido en un array. Ejemplo:

```
int[] arr = new int[5];
int value = arr[10]; // Esto lanzará ArrayIndexOutOfBoundsException
```

3. ClassCastException: Se produce cuando intentas convertir un objeto a un tipo incompatible. Ejemplo:

```
Object obj = "Hola";
Integer num = (Integer) obj; // Esto lanzará ClassCastException
```

4. ArithmeticException: Se produce cuando ocurre un error aritmético, como dividir por cero. Ejemplo:

```
int result = 10 / 0; // Esto lanzará ArithmeticException
```

5. IllegalArgumentException: Se produce cuando un método recibe un argumento ilegal. Ejemplo:

```
DateFormat df = new SimpleDateFormat("dd/MM/yyyy");
df.parse("2024-05-07"); // Esto lanzará IllegalArgumentException
```

6. IllegalStateException: Se produce cuando el estado de un objeto no es adecuado para una operación determinada. Ejemplo:

```
Thread thread = new Thread();
thread.start();
thread.start(); // Esto lanzará IllegalStateException
```

7. IndexOutOfBoundsException: Se produce cuando intentas acceder a un índice fuera del rango permitido. Ejemplo:

```
List<Integer> list = new ArrayList<>();
Integer num = list.get(10); // Esto lanzará
IndexOutOfBoundsException
```

8. NoSuchElementException: Se produce cuando intentas acceder a un elemento que no existe. Ejemplo:

```
Scanner scanner = new Scanner(System.in);
int num = scanner.nextInt(); // Si no hay entrada, esto lanzará
NoSuchElementException
```

9. ConcurrentModificationException: Se produce cuando se modifica una estructura de datos de manera concurrente (por ejemplo, iterar sobre una lista mientras se modifica). Ejemplo:

```
List<Integer> list = new ArrayList<>();
for (Integer num : list) {
    list.add(num + 1); // Esto lanzará
ConcurrentModificationException
}
```

10. UnsupportedOperationException: Se produce cuando se llama a un método que no está soportado por la implementación. Ejemplo:

```
List<Integer> list = Collections.singletonList(1);
list.add(2); // Esto lanzará UnsupportedOperationException
```

11. NumberFormatException: Se produce cuando intentas convertir una cadena a un número, pero la cadena no tiene el formato adecuado. Ejemplo:

```
jString str = "abc";
int num = Integer.parseInt(str); // Esto lanzará
NumberFormatException
```

12. StringIndexOutOfBoundsException: Se produce cuando intentas acceder a un índice que está fuera del rango válido en una cadena. Ejemplo:

```
String str = "hello";
char ch = str.charAt(10); // Esto lanzará
StringIndexOutOfBoundsException
```

13. IllegalThreadStateException: Se produce cuando se intenta realizar una operación en un hilo en un estado ilegal para esa operación. Ejemplo:

```
Thread thread = new Thread();
thread.start();
thread.start(); // Esto lanzará IllegalThreadStateException
```

14. IllegalAccessException: Se produce cuando se intenta acceder a un miembro (como un campo o método) de una clase desde fuera de ella, pero ese miembro está marcado como privado o protegido y no es accesible. Ejemplo:

```
class MyClass {
    private int num;
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        int value = obj.num; // Esto lanzará IllegalAccessException
    }
}
```

15. NoSuchMethodException: Se produce cuando intentas acceder a un método que no existe en una clase. Ejemplo:

```
class MyClass {
    public void myMethod() {
        System.out.println("Hola");
    }
}

public class Main {
    public static void main(String[] args) throws
NoSuchMethodException {
        MyClass obj = new MyClass();
        obj.nonExistentMethod(); // Esto lanzará
NoSuchMethodException
    }
}
```

16. NoSuchFieldException: Se produce cuando intentas acceder a un campo que no existe en una clase. Ejemplo:

```
class MyClass {
    public int myField;
}

public class Main {
    public static void main(String[] args) throws
NoSuchFieldException {
        MyClass obj = new MyClass();
        int value = obj.nonExistentField; // Esto lanzará
NoSuchFieldException
    }
```

```
}
```

17. ClassNotFoundException: Se produce cuando intentas cargar una clase que no existe. Ejemplo:

```
public class Main {
    public static void main(String[] args) throws
ClassNotFoundException {
        Class<?> cls = Class.forName("NonExistentClass"); // Esto
lanzará ClassNotFoundException
    }
}
```

18. InstantiationException: Se produce cuando intentas instanciar una clase abstracta o una interfaz, o cuando intentas instanciar una clase mediante un constructor que no está accesible. Ejemplo:

```
abstract class AbstractClass {}

public class Main {
    public static void main(String[] args) throws
InstantiationException {
        AbstractClass obj = AbstractClass.class.newInstance(); //
Esto lanzará InstantiationException
    }
}
```

19. InterruptedException: Se produce cuando un hilo está esperando, durmiendo o en estado de bloqueo, y otro hilo lo interrumpe. Ejemplo:

```
public class Main {
    public static void main(String[] args) {
        Thread.currentThread().interrupt();
        try {
            Thread.sleep(1000); // Esto lanzará InterruptedException
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

20. NoSuchAlgorithmException: Se produce cuando se solicita un algoritmo de cifrado que no está disponible en el entorno de ejecución. Ejemplo:

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

```
NoSuchAlgorithmException {
        MessageDigest md = MessageDigest.getInstance("SHA-256"); //
Esto lanzará NoSuchAlgorithmException
    }
}
```

21. IOException: Esta excepción es la clase base para excepciones producidas por operaciones de E/S (entrada/salida) fallidas o interrumpidas. Ejemplo:

```
try {
    FileReader fileReader = new FileReader("archivo.txt");
    // Operaciones de lectura
} catch (IOException e) {
    e.printStackTrace();
}
```

22. FileNotFoundException: Se produce cuando intentas acceder a un archivo que no existe en el sistema de archivos. Ejemplo:

```
try {
    FileInputStream fileInputStream = new
FileInputStream("archivo.txt");
    // Operaciones de lectura
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

23. EOFException: Se produce cuando se alcanza el final de un flujo de entrada (stream) antes de que se haya leído todo lo esperado. Ejemplo:

```
try {
    DataInputStream dataInputStream = new DataInputStream(new
FileInputStream("datos.bin"));
    while (true) {
        int value = dataInputStream.readInt();
        // Operaciones de lectura
    }
} catch (EOFException e) {
    e.printStackTrace();
}
```

24. SocketException: Se produce cuando ocurre un error de socket durante una operación de red. Ejemplo:

```
try {
    Socket socket = new Socket("localhost", 8080);
    // Operaciones de comunicación
} catch (SocketException e) {
    e.printStackTrace();
}
```

25. ConnectException: Se produce cuando no se puede establecer una conexión a un servidor remoto. Ejemplo:

```
try {
    Socket socket = new Socket("servidor_remoto.com", 8080);
    // Operaciones de comunicación
} catch (ConnectException e) {
    e.printStackTrace();
}
```

26. UnknownHostException: Se produce cuando no se puede resolver el nombre de host durante una operación de red. Ejemplo:

```
try {
    Socket socket = new Socket("host_desconocido.com", 8080);
    // Operaciones de comunicación
} catch (UnknownHostException e) {
    e.printStackTrace();
}
```

27. ProtocolException: Se produce cuando se encuentra un error en el protocolo utilizado en una comunicación de red. Ejemplo:

```
try {
    URL url = new URL("http://ejemplo.com");
    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
    connection.setRequestMethod("INVALID_METHOD"); // Esto lanzará
ProtocolException
} catch (ProtocolException e) {
    e.printStackTrace();
}
```

28. MalformedURLException: Se produce cuando intentas crear una URL con un formato incorrecto. Ejemplo:

```
try {
    URL url = new URL("esto_no_es_una_url");
} catch (MalformedURLException e) {
```

```
    e.printStackTrace();
}
```

29. URISyntaxException: Se produce cuando se encuentra un error de sintaxis en una URI (Identificador de Recurso Uniforme). Ejemplo:

```
try {
    URI uri = new URI("esto_no_es_una_uri");
} catch (URISyntaxException e) {
    e.printStackTrace();
}
```

30. ParseException: Se produce cuando se encuentra un error al analizar una cadena en un formato específico (por ejemplo, fecha, hora, número, etc.). Ejemplo:

```
try {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    Date date = sdf.parse("fecha_invalida");
} catch (ParseException e) {
    e.printStackTrace();
}
```

31. DateTimeParseException: Se produce cuando ocurre un error al analizar una cadena en un objeto de fecha y hora. Ejemplo:

```
try {
    LocalDateTime dateTime = LocalDateTime.parse("2024-05-
07T10:15:30");
} catch (DateTimeParseException e) {
    e.printStackTrace();
}
```

32. DateTimeException: Se produce cuando ocurre un error relacionado con las operaciones de fecha y hora. Ejemplo:

```
try {
    LocalDate date = LocalDate.of(2024, 13, 45); // Esto lanzará
DateTimeException
} catch (DateTimeException e) {
    e.printStackTrace();
}
```

33. UnsupportedCharsetException: Se produce cuando intentas usar un conjunto de caracteres no admitido. Ejemplo:

```
try {
    Charset charset = Charset.forName("UTF-16LE");
} catch (UnsupportedCharsetException e) {
    e.printStackTrace();
}
```

34. CharacterCodingException: Se produce cuando ocurre un error durante la codificación o decodificación de caracteres. Ejemplo:

```
try {
    Charset charset = Charset.forName("UTF-8");
    CharsetEncoder encoder = charset.newEncoder();
    CharBuffer charBuffer = CharBuffer.allocate(1);
    ByteBuffer byteBuffer = encoder.encode(charBuffer); // Esto
lanzará CharacterCodingException
} catch (CharacterCodingException e) {
    e.printStackTrace();
}
```

35. SQLDataException: Se produce cuando ocurre un error relacionado con los datos en una operación SQL. Ejemplo:

```
try {
    int intValue = Integer.parseInt("abc"); // Esto lanzará
NumberFormatException, que puede convertirse en SQLDataException en
ciertos contextos
} catch (SQLDataException e) {
    e.printStackTrace();
}
```

36. SQLException: Se produce cuando ocurre un error en una operación SQL. Ejemplo:

```
try {
    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/database",
"usuario", "contraseña");
    Statement statement = connection.createStatement();
    statement.executeQuery("SELECT * FROM tabla_invalida"); // Esto
lanzará SQLException
} catch (SQLException e) {
    e.printStackTrace();
```

```
}
```

37. SQLNonTransientException: Se produce cuando ocurre un error SQL que no es transitorio, es decir, no se espera que se resuelva automáticamente con una nueva operación. Ejemplo:

```
try {
    throw new SQLNonTransientException("Error no transitorio");
} catch (SQLNonTransientException e) {
    e.printStackTrace();
}
```

38. SQLTransientException: Se produce cuando ocurre un error SQL que es transitorio, es decir, se espera que se resuelva automáticamente con una nueva operación. Ejemplo:

```
try {
    throw new SQLTransientException("Error transitorio");
} catch (SQLTransientException e) {
    e.printStackTrace();
}
```

39. SQLFeatureNotSupportedException: Se produce cuando se intenta usar una característica no admitida por el motor de base de datos. Ejemplo:

```
try {
    throw new SQLFeatureNotSupportedException("Característica no
soportada");
} catch (SQLFeatureNotSupportedException e) {
    e.printStackTrace();
}
```

40. SQLIntegrityConstraintViolationException: Se produce cuando se viola una restricción de integridad en una base de datos (por ejemplo, clave primaria duplicada). Ejemplo:

```
try {
    throw new SQLIntegrityConstraintViolationException("Violación de
restricción de integridad");
} catch (SQLIntegrityConstraintViolationException e) {
    e.printStackTrace();
}
```

41. SQLInvalidAuthorizationSpecException: Se produce cuando se intenta establecer una conexión a una base de datos con credenciales de acceso no válidas. Ejemplo:

```
try {
    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/database",
"usuario_incorrecto", "contraseña_incorrecta");
} catch (SQLInvalidAuthorizationSpecException e) {
    e.printStackTrace();
}
```

42. SQLSyntaxErrorException: Se produce cuando se encuentra un error de sintaxis en una sentencia SQL. Ejemplo:

```
try {
    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/database",
"usuario", "contraseña");
    Statement statement = connection.createStatement();
    statement.executeQuery("SELECT * FROM tabla_invalida"); // Esto
lanzará SQLSyntaxErrorException
} catch (SQLSyntaxErrorException e) {
    e.printStackTrace();
}
```

43. SQLTimeoutException: Se produce cuando una operación SQL excede el tiempo de espera especificado. Ejemplo:

```
try {
    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/database",
"usuario", "contraseña");
    Statement statement = connection.createStatement();
    statement.setQueryTimeout(1);
    statement.executeQuery("SELECT * FROM
tabla_con_muchos_registros"); // Si la consulta demora más de 1
segundo, lanzará SQLTimeoutException
} catch (SQLTimeoutException e) {
    e.printStackTrace();
}
```

44. SQLTransactionRollbackException: Se produce cuando una transacción SQL se revierte (rollback). Ejemplo:

```
try {
    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/database",
"usuario", "contraseña");
```

```
    connection.setAutoCommit(false);
    // Realizar operaciones de actualización o inserción
    connection.rollback(); // Esto lanzará
SQLTransactionRollbackException
} catch (SQLTransactionRollbackException e) {
    e.printStackTrace();
}
```

45. SQLTransientConnectionException: Se produce cuando ocurre un error de conexión de manera transitoria, es decir, se espera que la conexión se restablezca automáticamente. Ejemplo:

```
try {
    throw new SQLTransientConnectionException("Error transitorio de
conexión");
} catch (SQLTransientConnectionException e) {
    e.printStackTrace();
}
```

46. SQLNonTransientConnectionException: Se produce cuando ocurre un error de conexión que no es transitorio, es decir, no se espera que la conexión se restablezca automáticamente. Ejemplo:

```
try {
    throw new SQLNonTransientConnectionException("Error no
transitorio de conexión");
} catch (SQLNonTransientConnectionException e) {
    e.printStackTrace();
}
```

47. SQLRecoverableException: Se produce cuando se produce un error de base de datos que es recuperable, es decir, se puede intentar la operación nuevamente. Ejemplo:

```
try {
    throw new SQLRecoverableException("Error de base de datos
recuperable");
} catch (SQLRecoverableException e) {
    e.printStackTrace();
}
```

48. SQLWarning: Se produce cuando se emite una advertencia relacionada con una operación SQL, como un problema potencial pero no fatal. Ejemplo:

```
try {
    Connection connection =
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/database",
"usuario", "contraseña");
    Statement statement = connection.createStatement();
    statement.executeQuery("SELECT * FROM
tabla_con_datos_obsoletos"); // Esto puede lanzar una SQLWarning
    SQLWarning warning = statement.getWarnings();
    if (warning != null) {
        System.out.println("Advertencia: " + warning.getMessage());
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

49. ClosedChannelException: Se produce cuando intentas realizar una operación en un canal que ya ha sido cerrado. Ejemplo:

```
try {
    FileInputStream fis = new FileInputStream("archivo.txt");
    fis.close();
    fis.read(); // Esto lanzará ClosedChannelException
} catch (ClosedChannelException e) {
    e.printStackTrace();
}
```

50. NonReadableChannelException: Se produce cuando intentas leer desde un canal que no es legible. Ejemplo:

```
try {
    FileOutputStream fos = new FileOutputStream("archivo.txt");
    FileChannel channel = fos.getChannel();
    ByteBuffer buffer = ByteBuffer.allocate(1024);
    channel.read(buffer); // Esto lanzará
NonReadableChannelException
} catch (NonReadableChannelException e) {
    e.printStackTrace();
}
```

51. NonWritableChannelException: Se produce cuando intentas escribir en un canal que no es escribible. Ejemplo:

```
try {
    FileInputStream fis = new FileInputStream("archivo.txt");
    FileChannel channel = fis.getChannel();
    ByteBuffer buffer = ByteBuffer.allocate(1024);
```

```
    channel.write(buffer); // Esto lanzará
NonWritableChannelException
} catch (NonWritableChannelException e) {
    e.printStackTrace();
}
```

52. UnsupportedAddressTypeException: Se produce cuando se intenta usar un tipo de dirección no admitido por una operación de socket. Ejemplo:

```
try {
    InetAddress address = InetAddress.getByAddress(new byte[]{127,
0, 0, 1});
    ServerSocketChannel serverSocketChannel =
ServerSocketChannel.open();
    serverSocketChannel.bind(new InetSocketAddress(address, 8080));
// Esto lanzará UnsupportedAddressTypeException
} catch (UnsupportedAddressTypeException | IOException e) {
    e.printStackTrace();
}
```

53. UnsupportedProtocolFamilyException: Se produce cuando se intenta usar una familia de protocolos no admitida. Ejemplo:

```
try {
    DatagramChannel datagramChannel = DatagramChannel.open();
    datagramChannel.bind(new InetSocketAddress(8080));
    datagramChannel.connect(new InetSocketAddress("localhost",
8080)); // Esto lanzará UnsupportedProtocolFamilyException
} catch (UnsupportedProtocolFamilyException | IOException e) {
    e.printStackTrace();
}
```

54. AlreadyBoundException: Se produce cuando se intenta vincular un socket a una dirección y puerto que ya están en uso. Ejemplo:

```
try {
    ServerSocket serverSocket1 = new ServerSocket(8080);
    ServerSocket serverSocket2 = new ServerSocket(8080); // Esto
lanzará AlreadyBoundException
} catch (IOException e) {
    e.printStackTrace();
}
```

55. UnsupportedCharsetException: Se produce cuando intentas usar un conjunto de caracteres no admitido. Ejemplo:

```
try {
    Charset charset = Charset.forName("ISO-8859-10"); // Esto
lanzará UnsupportedCharsetException
} catch (UnsupportedCharsetException e) {
    e.printStackTrace();
}
```

56. UnresolvedAddressException: Se produce cuando no se puede resolver una dirección de red. Ejemplo:

```
try {
    InetAddress address =
InetAddress.getByName("host_no_existente.com"); // Esto lanzará
UnresolvedAddressException
} catch (UnresolvedAddressException | UnknownHostException e) {
    e.printStackTrace();
}
```

57. UnsupportedEncodingException: Se produce cuando se intenta usar una codificación de caracteres no admitida. Ejemplo:

```
try {
    String str = "texto";
    byte[] bytes = str.getBytes("UTF-16LE"); // Esto lanzará
UnsupportedEncodingException
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
```

58. UnknownServiceException: Se produce cuando se intenta usar un servicio desconocido. Ejemplo:

```
try {
    Socket socket = new Socket("localhost", 8080);
    socket.setOption(StandardSocketOptions.IP_MULTICAST_TTL, 2); //
Esto lanzará UnknownServiceException
} catch (UnknownServiceException | IOException e) {
    e.printStackTrace();
}
```

59. InvalidKeyException: Se produce cuando se encuentra una clave no válida durante una operación de cifrado o firma. Ejemplo:

```
try {
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    SecretKeySpec keySpec = new SecretKeySpec(new byte[16], "AES");
```

```
    cipher.init(Cipher.ENCRYPT_MODE, keySpec); // Esto lanzará
InvalidKeyException
} catch (InvalidKeyException | NoSuchAlgorithmException |
NoSuchPaddingException e) {
    e.printStackTrace();
}
```

60. NoSuchAlgorithmException: Se produce cuando se solicita un algoritmo que no está disponible en el entorno de ejecución. Ejemplo:

```
try {
    MessageDigest md = MessageDigest.getInstance("SHA-512"); // Esto
lanzará NoSuchAlgorithmException
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

61. NoSuchPaddingException: Se produce cuando se intenta utilizar un esquema de relleno (padding) que no está disponible en la implementación de cifrado. Ejemplo:

```
try {
    Cipher cipher = Cipher.getInstance("AES/InvalidPadding"); //
Esto lanzará NoSuchPaddingException
} catch (NoSuchPaddingException | NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

62. ShortBufferException: Se produce cuando un buffer no tiene suficiente espacio para almacenar el resultado de una operación de cifrado o descifrado. Ejemplo:

```
try {
    ByteBuffer inputBuffer = ByteBuffer.allocate(16);
    ByteBuffer outputBuffer = ByteBuffer.allocate(8);
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
    cipher.encrypt(inputBuffer, outputBuffer, true); // Esto lanzará
ShortBufferException
} catch (ShortBufferException | NoSuchAlgorithmException |
NoSuchPaddingException e) {
    e.printStackTrace();
}
```

63. InvalidAlgorithmParameterException: Se produce cuando se proporciona un parámetro inválido a un algoritmo de cifrado o firma. Ejemplo:

```
try {
    KeyGenerator keyGen = KeyGenerator.getInstance("AES");
    SecureRandom random = new SecureRandom();
    keyGen.init(128, random); // Esto lanzará
InvalidAlgorithmParameterException
} catch (InvalidAlgorithmParameterException |
NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

64. IllegalBlockSizeException: Se produce cuando el tamaño de los datos de entrada no es válido para el algoritmo de cifrado. Ejemplo:

```
try {
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] input = new byte[15];
    byte[] output = cipher.doFinal(input); // Esto lanzará
IllegalBlockSizeException
} catch (IllegalBlockSizeException | NoSuchAlgorithmException |
NoSuchPaddingException | InvalidKeyException | BadPaddingException
e) {
    e.printStackTrace();
}
```

65. BadPaddingException: Se produce cuando ocurre un error de relleno durante la operación de descifrado. Ejemplo:

```
try {
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] input = new byte[16];
    byte[] output = cipher.doFinal(input); // Esto lanzará
BadPaddingException
} catch (BadPaddingException | NoSuchAlgorithmException |
NoSuchPaddingException | InvalidKeyException |
IllegalBlockSizeException e) {
    e.printStackTrace();
}
```

66. DigestException: Se produce cuando ocurre un error durante la generación o verificación de un resumen (hash) de mensaje. Ejemplo:

```
try {
    MessageDigest digest = MessageDigest.getInstance("SHA-256");
```

```
    byte[] input = new byte[16];
    byte[] output = digest.digest(input); // Esto lanzará
DigestException
} catch (DigestException | NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

67. SignatureException: Se produce cuando ocurre un error durante la generación o verificación de una firma digital. Ejemplo:

```
try {
    Signature signature = Signature.getInstance("SHA256withRSA");
    signature.initVerify(publicKey);
    byte[] data = new byte[16];
    signature.update(data);
    boolean valid = signature.verify(signatureBytes); // Esto
lanzará SignatureException
} catch (SignatureException | NoSuchAlgorithmException |
InvalidKeyException e) {
    e.printStackTrace();
}
```

68. KeyException: Se produce cuando ocurre un error relacionado con una clave criptográfica, como una clave no válida. Ejemplo:

```
try {
    KeyGenerator keyGen = KeyGenerator.getInstance("AES");
    keyGen.init(128); // Esto lanzará KeyException
} catch (KeyException | NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

69. InvalidKeySpecException: Se produce cuando se proporciona una especificación de clave inválida. Ejemplo:

```
try {
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(new
byte[64]);
    PublicKey publicKey = keyFactory.generatePublic(keySpec); //
Esto lanzará InvalidKeySpecException
} catch (InvalidKeySpecException | NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

70. InvalidParameterSpecException: Se produce cuando se proporciona una especificación de parámetro inválida a un algoritmo criptográfico. Ejemplo:

```
try {
    AlgorithmParameterSpec paramSpec = new IvParameterSpec(new
byte[8]);
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec); // Esto
lanzará InvalidParameterSpecException
} catch (InvalidParameterSpecException | NoSuchAlgorithmException |
NoSuchPaddingException | InvalidKeyException e) {
    e.printStackTrace();
}
```

71. CertificateException: Se produce cuando ocurre un error relacionado con certificados en operaciones de seguridad. Ejemplo:

```
try {
    CertificateFactory factory =
CertificateFactory.getInstance("X.509");
    Certificate cert = factory.generateCertificate(inputStream); //
Esto puede lanzar CertificateException
} catch (CertificateException e) {
    e.printStackTrace();
}
```

72. CertificateEncodingException: Se produce cuando ocurre un error de codificación o decodificación de un certificado. Ejemplo:

```
try {
    CertificateFactory factory =
CertificateFactory.getInstance("X.509");
    X509Certificate cert = (X509Certificate)
factory.generateCertificate(inputStream);
    byte[] encoded = cert.getEncoded(); // Esto puede lanzar
CertificateEncodingException
} catch (CertificateEncodingException | CertificateException e) {
    e.printStackTrace();
}
```

73. CertificateExpiredException: Se produce cuando se intenta utilizar un certificado que ha caducado. Ejemplo:

```
try {
    X509Certificate cert = (X509Certificate)
```

```
keystore.getCertificate("mi_certificado");
    cert.checkValidity(); // Esto puede lanzar
CertificateExpiredException
} catch (CertificateExpiredException | KeyStoreException e) {
    e.printStackTrace();
}
```

74. CertificateNotYetValidException: Se produce cuando se intenta utilizar un certificado que aún no es válido. Ejemplo:

```
try {
    X509Certificate cert = (X509Certificate)
keystore.getCertificate("mi_certificado");
    cert.checkValidity(); // Esto puede lanzar
CertificateNotYetValidException
} catch (CertificateNotYetValidException | KeyStoreException e) {
    e.printStackTrace();
}
```

75. CertPathBuilderException: Se produce cuando ocurre un error al construir un camino de certificación (CertPath). Ejemplo:

```
try {
    CertPathBuilder builder = CertPathBuilder.getInstance("PKIX");
    CertPath certPath = builder.build(certPathParameters); // Esto
puede lanzar CertPathBuilderException
} catch (CertPathBuilderException | NoSuchAlgorithmException |
InvalidAlgorithmParameterException e) {
    e.printStackTrace();
}
```

76. CertPathValidatorException: Se produce cuando ocurre un error al validar un camino de certificación (CertPath). Ejemplo:

```
try {
    CertPathValidator validator =
CertPathValidator.getInstance("PKIX");
    CertPath certPath = CertPath.getInstance("X.509");
    CertPathValidatorResult result = validator.validate(certPath,
params); // Esto puede lanzar CertPathValidatorException
} catch (CertPathValidatorException | NoSuchAlgorithmException |
InvalidAlgorithmParameterException | CertPathBuilderException e) {
    e.printStackTrace();
}
```

77. CRLException: Se produce cuando ocurre un error relacionado con la lista de revocación de certificados (CRL). Ejemplo:

```
try {
    CertificateFactory factory =
CertificateFactory.getInstance("X.509");
    X509CRL crl = (X509CRL) factory.generateCRL(inputStream); //
Esto puede lanzar CRLException
} catch (CRLException | CertificateException e) {
    e.printStackTrace();
}
```

78. CertStoreException: Se produce cuando ocurre un error relacionado con el almacenamiento de certificados. Ejemplo:

```
try {
    CertStore store = CertStore.getInstance("LDAP", new
LDAPCertStoreParameters());
    Collection<Certificate> certs = store.getCertificates(selector);
// Esto puede lanzar CertStoreException
} catch (CertStoreException e) {
    e.printStackTrace();
}
```

79. KeyStoreException: Se produce cuando ocurre un error relacionado con el almacenamiento de claves. Ejemplo:

```
try {
    KeyStore keystore =
KeyStore.getInstance(KeyStore.getDefaultType());
    keystore.load(null, null);
    keystore.store(outputStream, password); // Esto puede lanzar
KeyStoreException
} catch (KeyStoreException | IOException | NoSuchAlgorithmException
| CertificateException e) {
    e.printStackTrace();
}
```

80. InvalidKeyException: Se produce cuando ocurre un error relacionado con una clave, como una operación de cifrado o firma con una clave no válida. Ejemplo:

```
try {
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, key); // Esto puede lanzar
InvalidKeyException
} catch (InvalidKeyException | NoSuchAlgorithmException |
```

```
NoSuchPaddingException e) {
    e.printStackTrace();
}
```