National College of Ireland

Secure Programming for Application Development

Project tasks

National College of Ireland

REVISION

Encapsulation

National College of Ireland

- Protect the data: Data members should be **private**
  - You must have a really good reason for not making them private!!!
- Define well the interface of the object (**only the required operations should be public!!!**)

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

Example

National College of Ireland

```
public class BankAccount {
    // data members of BankAccount class
    String accountOwnerName;
    String accountOwnerSSN;
    int accountNumber;
    double balance;
    //…constructor and methods
}
```

Wrong! – break of encapsulation. Data is exposed!

Compliant solution

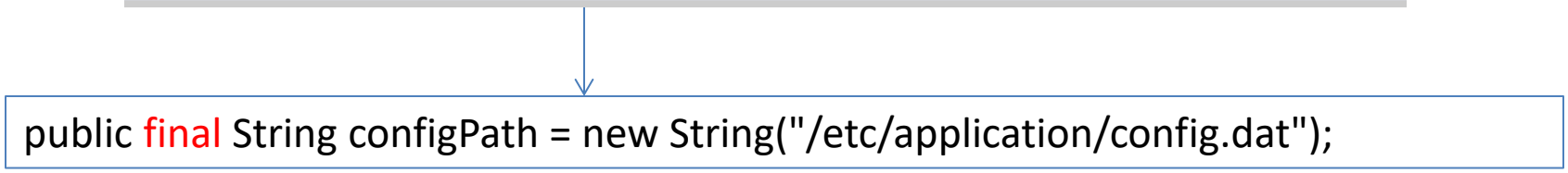
```
public class BankAccount {
    // data members of BankAccount class
    private String accountOwnerName;
    private String accountOwnerSSN;
    private int accountNumber;
    private double balance;
    //…constructor and methods
}
```

final keyword

National College of Ireland

- Make classes final
  - If a class isn't final, an attacker could try to extend it in a dangerous and unforeseen way.
- Sensitive public data (e.g. a path to a config file) should be final to avoid any modification

```
public String configPath = new String("/etc/application/config.dat");
```



```
public final String configPath = new String("/etc/application/config.dat");
```

https://cwe.mitre.org/data/definitions/493.html

Avoid **CWE-396: Declaration of Catch for Generic Exception**

National College of Ireland

yes

```
try {
  doExchange();

}
catch (IOException e) {
  logger.error("doExchange failed", e);

}
catch (InvocationTargetException e) {

  logger.error("doExchange failed", e);

}
catch (SQLException e) {

  logger.error("doExchange failed", e);

}
```

```
try {
  doExchange();

}
catch (Exception e) {
  logger.error("doExchange failed", e);

}
```

no

https://cwe.mitre.org/data/definitions/396.html

Avoid **CWE-396: Declaration of Catch for Generic Exception**

National College of Ireland

- **Example from the training project – UsersDao.java**

CWE-396 present here

```
public static int AddPublisher( String Publisher)
{
    int status= 0;
    try(Connection con = DB.getConnection()) {
            PreparedStatement ps=con.prepareStatement("insert into Publisher(PublisherName) values(?)");
            ps.setString(1,Publisher);
            status=ps.executeUpdate();
            con.close();
    }catch(Exception e){System.out.println(e);}
    return status;

}
```

```
    public static int AddPublisher( String Publisher)
    {
        int status= 0;
        try(Connection con = DB.getConnection()) {
                PreparedStatement ps=con.prepareStatement("insert into Publisher(PublisherName) values(?)");
                ps.setString(1,Publisher);
                status=ps.executeUpdate();
                con.close();
        }catch(SQLException e){System.out.println(e);}
        return status;

    }
```

CWE-396 avoided

National College of Ireland

PROJECT TASKS

Hashing the Password

National College of Ireland

Algorithms

- **MD5**
- **SHA algorithms**
- **PBKDF2**
  - PBKDF2WithHmacSHA512
  - PBKDF2WithHmacSHA1
- **Bcrypt**
- **Scrypt**
- **Argon 2 -** the winner of the [Password Hashing Competition](), 2015

Implementation provided by Java

Implementation provided by external libraries only

Hashing the Password

National College of Ireland

Task 1

- Write a method that given a string (the password) returns the hashed value.
  - use an algorithm provided by Java (e.g. MD5 or any of the SHA)

  Starting point:

  - MessageDigest md = MessageDigest.getInstance("MD5");
  - MessageDigest md = MessageDigest.getInstance("SHA-256");
  - Etc.

https://docs.oracle.com/javase/8/docs/api/java/security/MessageDigest.html

…and Salting the Password

National College of Ireland

- SEI CERT Oracle Coding Standard for Java: **MSC02-J. Generate strong random numbers**
- Use SecureRandom
  - To randomly generate the salt

https://wiki.sei.cmu.edu/confluence/display/java/MSC02-J.+Generate+strong+random+numbers

…and Salting the Password

National College of Ireland

- SEI CERT Oracle Coding Standard for Java: **MSC02-J. Generate strong random numbers**
- Non-compliant code

```
import java.util.Random;
// ...

Random number = new Random(123L);
//...
for (int i = 0; i < 20; i++) {
  // Generate another random integer in the range [0, 20]
  int n = number.nextInt(21);
  System.out.println(n);
}
```

https://wiki.sei.cmu.edu/confluence/display/java/MSC02-J.+Generate+strong+random+numbers

…and Salting the Password

National College of Ireland

- SEI CERT Oracle Coding Standard for Java: **MSC02-J. Generate strong random numbers**
- Compliant code

```
import java.security.SecureRandom;
import java.security.NoSuchAlgorithmException;
// ...

public static void main (String args[]) {
  SecureRandom number = new SecureRandom();
  // Generate 20 integers 0..20
  for (int i = 0; i < 20; i++) {
    System.out.println(number.nextInt(21));
  }
}
```

https://wiki.sei.cmu.edu/confluence/display/java/MSC02-J.+Generate+strong+random+numbers

…and Salting the Password

National College of Ireland

- SEI CERT Oracle Coding Standard for Java: **MSC02-J. Generate strong random numbers**
- Compliant code … in Java 8

```
import java.security.SecureRandom;
import java.security.NoSuchAlgorithmException;
// ...

public static void main (String args[]) {
   try {
     SecureRandom number = SecureRandom.getInstanceStrong();
     // Generate 20 integers 0..20
     for (int i = 0; i < 20; i++) {
       System.out.println(number.nextInt(21));
     }
   } catch (NoSuchAlgorithmException nsae) {
     // Forward to handler
   }
}
```

https://wiki.sei.cmu.edu/confluence/display/java/MSC02-J.+Generate+strong+random+numbers

…and Salting the Password

National College of Ireland

Task 2

Write a method that returns the salt using SecureRandom.

Task 3

Write a method that generates a hashed and salted the password (combines Task 1 and Task 2)

Note: For a complete workflow in the project, the generated password should be stored in the database together with the salt.

Never hard code sensitive information

- SEI CERT Oracle Coding Standard for Java **MSC03-J. Never hard code sensitive information**

Task 4

- In DB.java the connection to the database info (e.g password) is stored in clear text
- Modify the implementation so that the sensitive info is stored encrypted
  - Use an encryption algorithm provided by Java platform
- Comment on how you are going to store the encryption key

https://wiki.sei.cmu.edu/confluence/display/java/MSC03-J.+Never+hard+code+sensitive+information

https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html

Other Resources

- https://www.veracode.com/blog/research/encryption-and-decryption-java-cryptography

https://wiki.sei.cmu.edu/confluence/display/java/MSC03-J.+Never+hard+code+sensitive+information

https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html