Devante Wilson 100554361
November 13th, 2015

Lab-4: Inheritance, Abstract Classes, Polymorphism, and Interfaces

1.

ASuperClass.java

```
/**
 * Devante Wilson
 * November 13, 2015
 *
 * Question 1
 * Superclass
 */

public class ASuperClass
{
    public ASuperClass()
    {
        System.out.println("In superclass constructor");
    }
}
```

ASubClass.java

```
/**
 * Devante Wilson
 * November 13, 2015
 *
 * Question 1
 * Subclass of SuperClass
 */
public class ASubClass extends ASuperClass
{
    public ASubClass()
    {
        System.out.println("In subclass constructor");
    }
}
```

DemoConstructors.java

```
/**
 * Devante Wilson
 * November 13, 2015
 *
 * Question 1
 * DemoConstructor class
 */

public class DemoConstructors
{
    public static void main(String[] args)
    {
        ASubClass child = new ASubClass();
    }
}
```

1.1

```
In superclass constructor
In subclass constructor
```

Devante Wilson 100554361
November 13th, 2015

1.2

When using inheritance and extending a subclass from a superclass, the no-parameter constructor of the superclass is automatically invoked upon creating any objects; even an object of a subclass. In other words, even if the object "child" is an object of the subclass, the superclass constructor is still called. Furthermore, the same result would occur if **ASuperClass child = new ASubClass();**

2.

BankAccount.java | SavingsAccount.java | CheckingAccount.java

```
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Question 2
 * Superclass with basic
 * deposit and withdrawal methods
 */

public class BankAccount
{
    // instance variables
    String owner;
    Dollars balance;

    // method to deposit money
    void deposit(Dollars amount)
    {

    }

    // method to make withdrawals
    void withdrawal(Dollars amount)
    {

    }
}
```

BankAccount.java | SavingsAccount.java | CheckingAccount.java

```
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Question 2
 * Subclass of BankAccount class
 * Modeling a savings account.
 */

public class SavingsAccount extends BankAccount
{
    // instance variables
    Percentage annualInterestRate;

    // add a monthly interest deposit to account balance
    void depositMonthlyInterest()
    {

    }

    // method to make withdrawals
    void withdrawal(Dollars amount)
    {

    }
}
```

Devante Wilson 100554361
November 13th, 2015

BankAccount.java | SavingsAccount.java | CheckingAccount.java

```java
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Question 2
 * Subclass of BankAccount class
 * Modeling a checking account.
 */

public class CheckingAccount extends BankAccount
{
    // instance variable(s)
    Dollars insufficientFundsFee;

    // method to process a cheque
    void processCheck(Check checkToProcess)
    {

    }

    // method to process
    void withdrawal(Dollars amount)
    {

    }
}
```

3.

```java
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Question 3
 */

public class BaseballPlayer
{
    // instance variables
    private int jNum;
    private double battingAvg;

    public static void showOrigins()
    {
    System.out.println ("Abner Douleday is often" +
            "credited w/ Inventing baseball");
    }
}
```

BankAccount.java | SavingsAccount.java | CheckingAccount.java | BaseballPlayer.java | ProfessionalBaseball.java

```java
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Question 3
 */

public class ProfessionalBaseball extends BaseballPlayer
{
    // instance variable(s)
    double salary;

    public void showOrigins()
    {
        super.sh
        System.o                                    r league was in 1871");
    }
}
```

This instance method cannot override the static method from BaseballPlayer

1 quick fix available:

Remove 'static' modifier of 'BaseballPlayer.showOrigins'(..)

Press 'F2' for focus

In Java, a subclass cannot override static methods that are declared in its Super class.

A subclass cannot override a class method (a static method, a method you use without instantiating an object).

The showOrigins() method is used three times (defined in the superclass, defined in the subclass, and called in the subclass) because the programmer wanted to differentiate between when the sport of baseball was invented (in the superclass) and when the professional league was invented (in the subclass).

Devante Wilson

100554361
November 13th, 2015

4.

4.1

```
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Question 3
 * Abstract class Animal
 */

public abstract class Animal
{
    // instance variables
    private String name;

    // abstract method
    public abstract void speak();

    // accessor method
    public String getName()
    {
        return name;
    }

    // mutator method
    public void setName(String animalName)
    {
        name = animalName;
    }
}
```

4.2

```
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Subclass of superclass Animal
 */

public class Dog extends Animal
{
    // overridden method from superclass
    public void speak()
    {
        System.out.println("Wooof!");
    }
}
```

4.3

```
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Subclass of superclass Animal
 */

public class Cow extends Animal
{
    // overridden method from superclass
    public void speak()
    {
        System.out.println("Moo!");
    }
}
```

4.4

```
/**
 * Devante Wilson
 * November 15, 2015
 *
 * Tester class for Animal
 */

public class UseAnimal
{
    public static void main (String[] args)
    {
        // define objects
        Dog myDog = new Dog ();
        Cow myCow= new Cow ();

        // call methods and print results
        myDog.setName("Murphy");
        myCow.setName("Elsie");

        System.out.print(myDog.getName() + " says ");
        myDog.speak();
        System.out.print(myCow.getName() + " says ");
        myCow.speak();
    }
}
```

Console

<terminated> UseAnimal [Java App

```
Murphy says Wooof!
Elsie says Moo!
```