5COSC019W - Solutions to Tutorial 4 Exercises

1 Interfaces

```
interface Printable {
    void print();
}

class Letter implements Printable {
    private String text;

    Letter(String text) {
        this.text = text;
    }

    public void print() {
        System.out.println("text: " + text);
    }
}

class Student implements Printable {
    private String name;
    private String grades[] = new String[5];

    Student(String name, String grades[]) {
        this.name = name;

        int i = 0;
        // copy the first 5 elements of argument to instance field grades
        while (i < 5 && i < grades.length) {
            this.grades[i] = grades[i];
            ++i;
        }
    }

    public void print() {
        System.out.println("name: " + name);
        System.out.print("grades: ");
        for (String s : grades)
            System.out.print(s + " ");
```

```
        System.out.println();
    }
}

public class PrintTest {
    public static void main(String[] args) {
        String grades1[] = {"A", "B", "B", "A", "C"};
        Student s1 = new Student("John", grades1);
        s1.print();

        String grades2[] = new String[5];
        grades2[0] = "C";
        grades2[1] = "A";
        grades2[2] = "B";
        grades2[3] = "D";
        grades2[4] = "F";
        Student s2 = new Student("Helen", grades2);
        s2.print();

        Letter l1 = new Letter("myletter");
        l1.print();
    }
}
```

2 Members (data and methods) of Interfaces

The program has four problems:

1. Method `solve` in the interface `Equation` is declared `private` and it does not have a body. An interface can have only public members but in recent versions of Java you can also have private methods assuming you provide a body to them.
2. Method `printFormula` in the interface has an implementation. Methods within an interface do not have a body but they are implemented in classes implementing the interface (exception to that are private, static and default methods).
3. Because `LinearEquation` implements `Equation` is should implement all methods of `Equation`. However, `printFormula()` is not implemented in `LinearEquation`.
4. Inside `solve` of `LinearEquation` an attempt is made to change the value of field `numberOfVariables` inherited by the interface `Equation`.

   However, all fields inside an interface are `final`.

3 The `Comparable` Interface

1. ```
   public class BankAccount implements Comparable<BankAccount> {
       private double balance;
   ```
   ```

```
    public BankAccount(double balance) {
        this.balance = balance;
    }

    public double getBalance() {
        return balance;
    }

    /**
        Compares two bank accounts.
        @param other the other BankAccount
        @return 1 if this bank account has a greater balance than the other one,
        -1 if this bank account is has a smaller balance than the other one,
        and 0 if both bank accounts have the same balance
    */
    public int compareTo(BankAccount other) {
        if (balance > other.getBalance())
            return 1;
        else if (balance > other.getBalance())
            return -1;
        else  // same balance on both accounts
            return 0;
    }
}
```

2.
```
import java.util.ArrayList;
import java.util.Collections;

public class ComparableTest {
    public static void main(String[] args) {
        // create three different bank account objects
        BankAccount ba1 = new BankAccount(100.0);
        BankAccount ba2 = new BankAccount(50.0);
        BankAccount ba3 = new BankAccount(20.0);

        // put bank accounts into a list
        ArrayList<BankAccount> list = new ArrayList<BankAccount>();
        list.add(ba1);
        list.add(ba2);
        list.add(ba3);

        // call the library sort method
        Collections.sort(list);

        // print out the sorted list
        for (BankAccount b : list)
            System.out.println(b.getBalance());
    }
```

```
    }
```

4 The Comparator Interface

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class MyComparator implements Comparator<BankAccount> {
    public int compare(BankAccount b1, BankAccount b2) {
        if (b1.balance < b2.balance)
            return -1;
        else if (b1.balance > b2.balance)
            return 1;
        else
            return 0;
    }
}

class BankAccount {
    double balance;

    BankAccount(double bal) {
        balance = bal;
    }
}

class ComparatorExample {
    public static void main(String[] args) {
        BankAccount ba1 = new BankAccount(20000.0);
        BankAccount ba2 = new BankAccount(200.0);
        BankAccount ba3 = new BankAccount(550.0);

        // put bank accounts into a list
        ArrayList<BankAccount> list = new ArrayList<BankAccount>();
        list.add(ba1);
        list.add(ba2);
        list.add(ba3);

        // call the library sort method
        Collections.sort(list, new MyComparator());

        // print out the sorted list
        for (BankAccount b : list)
            System.out.println(b.balance);
    }
}
```

5 Abstract Classes

1. File `DriverLicense.java`:

```
import java.util.*;

public class DriverLicense extends Card {
    private int expirationYear;

    public boolean isExpired() {
        // find out the current year
        GregorianCalendar calendar = new GregorianCalendar();
        int current_year = calendar.get(Calendar.YEAR);

        if (expirationYear < current_year)
            return true;
        else
            return false;
    }
}
```

File `Passport.java`:

```
import java.util.*;

public class Passport extends Card {
    private String birthLocation;
    private int expirationYear;

    public boolean isExpired() {
        // find out the current year
        GregorianCalendar calendar = new GregorianCalendar();
        int current_year = calendar.get(Calendar.YEAR);

        if (expirationYear < current_year)
            return true;
        else
            return false;
    }
}
```

File `CreditCard.java`:

```
public class CreditCard extends Card {
    private int pinNumber;
    private int number;

    public boolean isExpired() {
        // assume credit cards never expire
        return false;
```

```
    }
}
```

2. The complete code for the above classes with the implemented constructors is shown below. Note that some more constructors would be necessary, if for example we would like to create a `Passport` object without passing any arguments to it.

   File `DriverLicense.java`:

```
import java.util.*;

public class DriverLicense extends Card {
    private int expirationYear;

    public DriverLicense(String n, int expire) {
        super(n);
        expirationYear = expire;
    }

    public DriverLicense(int expire) {
        expirationYear = expire;
    }

    public boolean isExpired() {
        // find out the current year
        GregorianCalendar calendar = new GregorianCalendar();
        int current_year = calendar.get(Calendar.YEAR);

        if (expirationYear < current_year)
            return true;
        else
            return false;
    }
}
```

   File `Passport.java`:

```
import java.util.*;

public class Passport extends Card {
    private String birthLocation;
    private int expirationYear;

    public Passport(String n, String birth, int expire) {
        super(n);
        birthLocation = birth;
        expirationYear = expire;
    }

    public Passport(String birth, int expire) {
        birthLocation = birth;
```

```
            expirationYear = expire;
        }

        public boolean isExpired() {
            // find out the current year
            GregorianCalendar calendar = new GregorianCalendar();
            int current_year = calendar.get(Calendar.YEAR);

            if (expirationYear < current_year)
                return true;
            else
                return false;
        }
    }
```

File `CreditCard.java`:

```
    public class CreditCard extends Card {
        private int pinNumber;
        private int number;

        public CreditCard(String n, int pin, int num) {
            super(n);
            pinNumber = pin;
            number = num;
        }

        public CreditCard(int pin, int num) {
            pinNumber = pin;
            number = num;
        }

        public boolean isExpired() {
            // assume credit cards never expire
            return false;
        }
    }
```

The following shows a test class which is used to test the implemented classes:

```
public class CardHierarchyTest {
    public static void main(String[] args) {
        DriverLicense d1 = new DriverLicense("John Smith", 2008);
        System.out.println("License of John expired: " + d1.isExpired());

        DriverLicense d2 = new DriverLicense("Bill Jones", 2004);
        System.out.println("License of Bill expired: " + d2.isExpired());

        Passport p1 = new Passport("John Smith", "London", 2012);
```

```
        System.out.println("Passport of John expired: " + p1.isExpired());

        Passport p2 = new Passport("Bill Jones", "Glasgow", 2020);
        System.out.println("Passport of Bill expired: " + p2.isExpired());

        CreditCard c1 = new CreditCard("John Smith", 5555, 444444444);
        // credit card without a name on it!
        CreditCard c2 = new CreditCard(7575, 1515151761);
    }
}
```

6 Overriding Methods

The classes are modified to include the overridden method and they are shown below.

File `DriverLicense.java`:

```
import java.util.*;

public class DriverLicense extends Card {
    private int expirationYear;

    public DriverLicense(String n, int expire) {
        super(n);
        expirationYear = expire;
    }

    public DriverLicense(int expire) {
        expirationYear = expire;
    }

    public boolean isExpired() {
        // find out the current year
        GregorianCalendar calendar = new GregorianCalendar();
        int current_year = calendar.get(Calendar.YEAR);

        if (expirationYear < current_year)
            return true;
        else
            return false;
    }

    public String format() {
        String name = super.format();
        return name + ", Expires: " + expirationYear;
    }
}
```

File `Passport.java`:

```
import java.util.*;

public class Passport extends Card {
    private String birthLocation;
    private int expirationYear;

    public Passport(String n, String birth, int expire) {
        super(n);
        birthLocation = birth;
        expirationYear = expire;
    }

    public Passport(String birth, int expire) {
        birthLocation = birth;
        expirationYear = expire;
    }

    public boolean isExpired() {
        // find out the current year
        GregorianCalendar calendar = new GregorianCalendar();
        int current_year = calendar.get(Calendar.YEAR);

        if (expirationYear < current_year)
            return true;
        else
            return false;
    }

    public String format() {
        String name = super.format();
        return name + ", Birth location: " + birthLocation + ", Expires: " +
                      expirationYear;
    }
}
```

File `CreditCard.java`:

```
public class CreditCard extends Card {
    private int pinNumber;
    private int number;

    public CreditCard(String n, int pin, int num) {
        super(n);
        pinNumber = pin;
        number = num;
    }

    public CreditCard(int pin, int num) {
        pinNumber = pin;
```

```
        number = num;
    }

    public boolean isExpired() {
        // assume credit cards never expire
        return false;
    }

    public String format() {
        String name = super.format();
        return name + ", pin: " + pinNumber + ", number: " + number;
    }
}
```

The following class tests the functionality of the implemented classes:

```
public class CardHierarchyTest {
    public static void main(String[] args) {
        DriverLicense d1 = new DriverLicense("John Smith", 2008);

        DriverLicense d2 = new DriverLicense("Bill Jones", 2004);

        Passport p1 = new Passport("John Smith", "London", 2012);

        Passport p2 = new Passport("Bill Jones", "Glasgow", 2020);

        CreditCard c1 = new CreditCard("John Smith", 5555, 444444444);
        // credit card without a name on it!
        CreditCard c2 = new CreditCard(7575, 1515151761);

        System.out.println(d1.format());
        System.out.println(d2.format());
        System.out.println(p1.format());
        System.out.println(p2.format());
        System.out.println(c1.format());
        System.out.println(c2.format());
    }
}
```

When `CardHierarchyTest` is run, it displays:

```
Card holder: John Smith, Expires: 2008
Card holder: Bill Jones, Expires: 2004
Card holder: John Smith, Birth location: London, Expires: 2012
Card holder: Bill Jones, Birth location: Glasgow, Expires: 2020
Card holder: John Smith, pin: 5555, number: 444444444
Card holder: , pin: 7575, number: 1515151761
```

7 Polymorphism

1. The output of the program is:

```
print A!
print B!
print C!
print D!
print D!
Exception in thread "main" java.lang.ClassCastException: A
        at PTester.main(PTester.java:45)
```

   As it is seen, an exception is generated in line 45.

2. Reference variable `b1` is of type `B` but points to an object of the subclass `D`. Therefore the cast is required, as otherwise an incompatible assignment is attempted.

3. Reference variable `a2` points to an object of class `A`.

   Therefore, during runtime the attempt to convert an object of type `A` to an object of type `D` will generate an exception.

8 Access Specifiers

Errors in lines 22, 28, 37, 43, 53 in `Program1.java`. `z` has private access in `P`

In `Program2`:

```
Program2.java:9: error: z has private access in P
        z = 4.5;  // line 10
        ^
Program2.java:10: error: w is not public in P; cannot be accessed from
outside package. It has the same package access
        w = 5.5;  // line 11
        ^
Program2.java:14: error: y has protected access in P
        p4.y = 3.5;  // line 15
          ^
Program2.java:15: error: z has private access in P
        p4.z = 4.5;  // line 16
          ^
Program2.java:16: error: w is not public in P; cannot be accessed from outside package
        p4.w = 5.5;  // line 17
          ^
Program2.java:24: error: y has protected access in P
        p5.y = 3.6;  // line 25
          ^
Program2.java:25: error: z has private access in P
        p5.z = 4.6;  // line 26
          ^
```

```
Program2.java:26: error: w is not public in P; cannot be accessed from outside package
        p5.w = 5.6;  // line 27
           ^
```

9 Challenge: A Program for Appointments

This is an optional challenge exercise. If you attempt this and if you have any doubts about your solution, you could show this to your tutor.