KIE1008: Programming 2
Week 5: Object-Oriented Programming – Inheritance

1. Consider the following class definition:

```
class circle
{
public
    void print() const;
    void setRadius(double);
    void setCenter(double, double);
    void getCenter(double&, double&);
    double getRadius();
    double area();
    circle();
    circle(double, double, double);

private:
    double xCoordinate;
    double yCoordinate;
    double radius;
};
```

```
class cylinder: public circle
{
public
    void print() const;
    void setHeight(double);
    double getHeight();
    double volume();
    double area();
    cylinder();
    cylinder(double, double,
             double, double);

    private:
        double height;
};
```

   a) Write the definitions of the member functions of the classes `circle` and `cylinder`.
   b) Identify the member functions of the class `cylinder` that overrides the member functions of the class `circle`.
   c) Suppose that you have the declaration

```
cylinder newCylinder;
```

   Determine the private members of the object `newCylinder`.

2. Consider the following class definitions:

```
class smart
{
public:
   void print() const;
   void set(int, int);
   int sum();
   smart();
   smart(int, int);
private:
   int x;
   int y;
   int secret();
};
```

```
class superSmart: public smart
{
public:
   void print() const;
   void set(int, int, int);
   int manipulate();
   superSmart();
   superSmart(int, int, int);
private:
    int z;
};
```

   Mark the following statements as valid or invalid. If a statement is invalid, explain why.

   a)
```
int smart::sum()
{
   return x + y + z;
}
```

   b)
```
void superSmart::set(int a, int b, int c)
{
   smart::set(a, b);
   z=c;
}
```

c)
```
int main()
{
   cout << superSmart.sum() << superSmart.z << endl;
}
```

3. Consider the following class definition:

```
class employee
{
public:
    void setData(string n = "", string d = "", int a = 0,
                 double p = 0);
    void setName(string n);
    string getName() const;
    void setDepartment(string dept);
    string getDepartment() const;
    void setAge(int a);
    int getAge() const;
    void setPay(double p);
    double getPay() const;

    employee(string n = "", string  d = "", int a = 0,
             double p = 0);

private:
    string name;
    string department;
    int age;
    double pay;
};
```

a) Identify and correct errors in the following class definition.

```
class hourlyEmployee: public class employee
{
public::
    void setData(string n = "", string d = "", int a = 0, double
                 p = 0, double hrsWk = 0, double payRate = 0.0);
    void setHoursWorked(double hrsWk) const;
    double getHoursWorked() const;
    void setHourlyPayRate(double payRate);
    double getHourlyPayRate() const;
    void setPay() const;
    hourlyEmployee(string n = "", string d = "", int a = 0, double
                   p = 0, double hrsWk = 0, double payRate = 0.0);

private;
    double hoursWorked;
    double hourlyPayRate;
};
```

b) Write the definition of the member functions of the class `hourlyEmployee` after correction in part (a). For function `setPay`, if `hoursworked >= 0` and `hourlyPayRate >= 0`, `pay = hoursworked * hourlyPayRate;` otherwise `pay = 0.0`.

c) Determine whether the function `setPay` of the class `hourlyEmployee` overrides or overloads the function `setPay` of the class `employee`.

4. Create an inheritance hierarchy that a bank might use to represent customers' bank accounts.
    a) Define the class `bankAccount` to store a bank customer's account number and balance. Suppose that account number is of type `int`, and balance is of type `double`. The class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it's greater than or equal to 0.0. If not, the balance should be set to 0.0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide at least four member functions as follows.
        i. Member function `credit` should add an amount to the current balance.
        ii. Member function `debit` should withdraw money from the `bankAccount` and ensure that the debit amount does not exceed the `bankAccount`'s balance. If it does, the balance should be left unchanged and the function should print the message "Debit amount exceeded account balance."
        iii. Member function `getBalance` should return the current balance.
        iv. Member function `printInfo` should print account information.
    b) Every bank offers a savings account. Derive the class `savingsAccount` from the class `bankAccount`. This class inherits members to store the account number and the balance from the base class, but also include a data member of type `double` indicating the interest rate (percentage) assigned to the `bankAccount`. `savingsAccount`'s constructor should receive the initial balance, as well as an initial value for the `savingsAccount`'s interest rate. `savingsAccount` should provide a public member function `calculateInterest` that indicates the amount of interest earned by an account. Class `savingsAccount` should redefine member functions `credit` and `debit` so that they subtract the fee from the account balance whenever either transaction is performed successfully.
    c) Write a program to test your classes.