İ.T.Ü.
Computer Engineering Department
Sanem SARIEL TALAY
Gülşen CEBİROĞLU ERYİĞİT

18.04.2011

Object Oriented Programming - 2nd Midterm Examination

St No	St Name	Signature	Q1	Q2	Total

Duration: 2 hours

Question 1: (50 Points) Some classes are to be designed for soccer player humanoid robot software. Assume that the robots can take two different **Planner** roles namely, **Attacker** and **Defender**.

- Each planner has the information on the **robotID** and the robot's **myX** and **myY** positions to plan effectively. **robotID** should be given a positive number, otherwise it is set to 0.
- Each planner can be **penalized** by setting the **robotID** , **myX** and **myY** values as -1.
- **Attacker** planner also possesses ball location information **ballX** and **ballY**.
- **Defender** planner also keeps track of information on a given **number of opponents** to defend and an opponent list to maintain IDs of these opponents. This list should be created in the dynamic memory as a **dynamic array** of integers. Each opponent ID is assigned as an incremented value of a previous value beginning with 0.
- In this design, you don't need to design actual motion plans for robots. You may just assume the **plan** methods of each planner just print special messages on the screen.
- **plan** method of **Defender** planner prints the message: "Robot id: <id> Defender number of opponents: <number>".
- **plan** method of **Attacker** planner prints the message: "Robot id: <id> Attacker ball loc: <x,y>".

You are asked to design these classes in C++ while **avoiding code repetition** as much as possible for all classes and ensuring **data hiding**. The following test code is given to guide the design of your classes. All data members must be **private**. Please write **all declarations and the bodies** of the required methods for the classes.

```
int main(){
    //Planner p1;   //COMPILER ERROR: No appropriate default constructor
    Planner p1(1);  //A planner is created with robotID 1 at robot loc (x,y): (0,0)
    //p1.printID();  //COMPILER ERROR: printID() prints "Robot id: <id>"
                      //but not accessible from outside
    Defender d1(2,1,1,3); //A defender is created with robotID 2
                          //at robot loc (x,y):(1,1) and
                          //the numOfOpponents 3 with IDs 0,1,2
    Defender d2(-1,2,2,2); //A defender is created with robotID 0 (-1 is not accepted)
                           //at robot loc (x,y): (2,2) and
                           //the numOfOpponents 2 with IDs 0,1
    Defender d3 = d2;
    d2.penalize();  //id,myX,myY values are set to -1
    d1 = d3;
    d3.penalize();  //id,myX,myY values are set to -1
    Attacker a1(4,5,5);  //An attacker is created with robotID 4 at
                         //robot loc(x,y): (0,0) and the ball position (x,y) at (5,5)
    char c;
    cout << "Is the ball close to the player?"; cin >> c;
    Planner *pptr;
    if (c == 'y')  //Attacker plan is selected
        pptr = &a1;  //"Robot id: <id> Attacker ball loc: <x,y>" is printed
    else  //Defender plan is selected
        pptr = &d1;  //"Robot id: <id> Defender num of opponents: <number>" is printed
    pptr->plan();
    return 0;
}
```

Question 2: (50 Points) Many statements in the following program cause **compile-time errors**. Please **comment out** (put // in front of) incorrect statements and **give the reason** (next to the erroneous line). After commenting out the incorrect statements, what will be the **output** of the program? Please write what will be displayed on the screen.

```
#include <iostream>
using namespace std;

class A{

private:
      int x;
public:
      int y;
protected:
      int z;

public:
      A(){cout<<"A created"<<endl;};
      ~A()
         {cout<<"A destructed"<<endl;};
      void f();
};

void A::f(){
      cout<<"A:"<<y<<endl;
}

class B:  public A{

public:
      int i;
      B(int,int,int);
      ~B()
        {cout<<"B destructed"<<endl;};
      void f();
};

B::B(int a, int b, int c){
      cout<<"B created"<<endl;
      x=a;
      y=b;
      z=c;
}

void B::f(){
      cout<<"B:"<<y<<endl;
}

class C:  private A{

public:
      int i;
      C(int,int,int);
      ~C()
        {cout<<"C destructed"<<endl;};
      void f();
};
```

```
C::C(int a,int b,int c){
      cout<<"C created"<<endl;
      x=a;
      y=b;
      z=c;
}

void C::f(){
      cout<<"C:"<<y<<endl;
}

void print(A *ptr){
      cout<<ptr->y<<endl;
}

int main(){
      B b(0,0,0);
      b.x=1;
      b.y=2;
      b.z=3;
      b.i=5;
      C c(0,0,0);
      c.x=1;
      c.y=2;
      c.z=3;
      c.i=5;
      A *aptr=&b;
      aptr->x=1;
      aptr->y=2;
      aptr->z=3;
      aptr->i=4;
      aptr=&c;
      aptr->f();
      print(aptr);
      return EXIT_SUCCESS;
 }
```

```
PROGRAM OUTPUT:
1.
2.
3.
4.
5.
6.
7.
8.
:
```

ANSWERS

Question 1:

```
class Planner{

    int robotID, myX, myY;

protected:
    void printID() const {cout << "Robot id: " << robotID;}

public:
    Planner(int id, int x = 0, int y = 0){
            if (id >= 0)
               robotID = id;
            else
               robotID = 0;
            myX = x;
            myY = y;
    }

    virtual void plan() const{
            cout <<"Default plan" << endl;
    }

    void penalize(){
            robotID = -1;
            myX = -1,
            myY = -1;
    }
};



class Attacker: public Planner{

    int ballX, ballY;

public:
    Attacker(int id, int bx, int by):Planner(id),ballX(bx),ballY(by){}

    void plan() const{
        Planner::printID();
        cout << " Attacker ball loc: " << ballX <<","<< ballY<< endl;
    }
};
```

```
class Defender: public Planner{

    int numOfOpponents;
    int *opponents;

public:
    Defender (int id, int x, int y, int num): Planner(id,x,y){
        numOfOpponents = num;
        opponents = new int[numOfOpponents];
        for (int i= 0; i< numOfOpponents;i++)
            opponents[i] = i;
    }

    Defender(const Defender &exObj):Planner(exObj){
        numOfOpponents = exObj.numOfOpponents;
        opponents = new int[numOfOpponents];

        for (int i= 0; i< numOfOpponents;i++)
            opponents[i] = exObj.opponents[i];

    }

    const Defender& operator=(const Defender& otherObj){
        Planner::operator=(otherObj);

        if(numOfOpponents != otherObj.numOfOpponents){
            delete [] opponents;
            numOfOpponents = otherObj.numOfOpponents;
            opponents = new int[numOfOpponents];
        }
        for (int i= 0; i< numOfOpponents;i++)
            opponents[i] = otherObj.opponents[i];

        return *this;
    }

    void plan() const{
        Planner::printID();
        cout << " Defender " << "number of opponents: " << numOfOpponents << endl;
    }

    ~Defender(){
        delete []opponents;
    }
};
```

Question 2:

```
#include <iostream>
using namespace std;

class A{

private:
      int x;
public:
      int y;
protected:
      int z;

public:
      A(){cout<<"A created"<<endl;};
      ~A()
         {cout<<"A destructed"<<endl;};
      void f();
};

void A::f(){
      cout<<"A:"<<y<<endl;
}

class B:  public A{

public:
      int i;
      B(int,int,int);
      ~B()
        {cout<<"B destructed"<<endl;};
      void f();
};

B::B(int a, int b, int c){
      cout<<"B created"<<endl;
      // x=a; x is private
      y=b;
      z=c;
}

void B::f(){
      cout<<"B:"<<y<<endl;
}

class C:  private A{

public:
      int i;
      C(int,int,int);
      ~C()
        {cout<<"C destructed"<<endl;};
      void f();
};
```

```
C::C(int a,int b,int c){
      cout<<"C created"<<endl;
      // x=a; x is private
      y=b;
      z=c;
}

void C::f(){
      cout<<"C:"<<y<<endl;
}

void print(A *ptr){
      cout<<ptr->y<<endl;
}

int main(){
      B b(0,0,0);
      // b.x=1; x is private
      b.y=2;
      // b.z=3; z is protected
      b.i=5;
      C c(0,0,0);
      // c.x=1; x is private
      // c.y=2; y is private due
      to private inheritance
      // c.z=3; z is protected
      c.i=5;
      A *aptr=&b;
      // aptr->x=1; x is private
      aptr->y=2;
      // aptr->z=3; z is protected
      // aptr->i=4; i is not a
      member of base class A
      // aptr=&c; A is an
      inaccessible base for C due
      to private inheritance
      aptr->f();
      print(aptr);
      return EXIT_SUCCESS;
 }
```

```
PROGRAM OUTPUT:
1. A created
2. B created
3. A created
4. C created
5. A:2
6. 2
7. C destructed
8. A destructed
9. B destructed
10.A destructed
```