Institut für Datentechnik
Fachgebiet Echtzeitsysteme
Prof. Dr. rer. nat. Andy Schürr

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Klausur zum Programmierpraktikum C und C++

14.10.2014

Code-Handout

Name: _____
Vorname: _____
Matrikelnummer: _____

Beachten Sie bitte Folgendes:

- **Dieses Handout besteht aus 6 einseitig bedruckten Blättern (inkl. Deckblatt).**

  Prüfen Sie bitte auf Vollständigkeit.

- **Das Handout muss zusammen mit der Klausur abgegeben werden.**

- **Obwohl Sie im Handout frei schreiben dürfen, werden jegliche Notizen und Kommentare im Handout ignoriert und auf keinen Fall bewertet.**

Szenario

Das durchgängige Beispiel dieser Klausur ist ein System zur Überwachung des Schiffsverkehrs in Küstennähe (`CoastControlSystem`).

Wir modellieren aktuell drei Arten von Schiffen: unbekannte Schiffe (Klasse `Ship`), Frachtschiffe (`CargoShip`) und Fähren (`Ferry`). Jedes Schiff hat dabei ein eindeutiges Rufzeichen (*call sign*), ein Symbol für die Darstellung auf dem Radarschirm (*marker*) und eine geografische Position nach Länge (*longitude*) und Breite (*latitude*).

Das `CoastControlSystem` erhält vom Radar-System (Klasse `Radar`) Informationen über Schiffe, die in den überwachten Bereich einfahren. Die vom `CoastControlSystem` genutzten Schiffsinstanzen werden mittels einer Factory (Klasse `ShipFactory`) erzeugt.

Quellcode

```
----------------------  main_exercise2.cpp  -------------------------

                [Dieser Code ist für Aufgabe 2 bestimmt.]
```

```
const std::string describe(const Ship &ship) {
    stringstream str;
    str << "Ship " << ship.getCallSign()
        << " shown as " << ship.getMarker()
        << ".";
    return str.str();
}



int main_exercise2() {

    // We do not use ShipFactory here, yet.

    Ship *ship = new Ship("SHIP", 53.667, 5.325);
    Ship *cargo = new CargoShip("CARGO", 54.667, 5.825);
    Ship *ferry = new Ferry("FERRY", 53.977, 6.825);

    cout << describe(*ship) << endl;
    cout << describe(*cargo) << endl;
    cout << describe(*ferry) << endl;

    delete ship;
    delete cargo;
    delete ferry;

    return 0;
}
```

```
 ---------------------------  Ship.h  ----------------------------
#ifndef SHIP_H_
#define SHIP_H_

#include <string>

class Ship {

public:
    Ship(const std::string &callSign,
         double latitude, double longitude);

    virtual ~Ship();

    // Unique for each ship
    const std::string getCallSign() const;

    // Symbol on the radar screen
    const std::string getMarker() const;

    double getLatitude() const;
    double getLongitude() const;

private:
    std::string callSign;

    double latitude, longitude;
};

#endif /* SHIP_H_ */
```

```
 --------------------------  Ship.cpp  ----------------------------
#include "Ship.h"

Ship::Ship(const std::string &callSign, double latitude,
           double longitude) :
      callSign(callSign), latitude(latitude), longitude(longitude) {}

Ship::~Ship() {}

const std::string Ship::getMarker() const {
    return "?";
}

const std::string Ship::getCallSign() const { return this->callSign; }

double Ship::getLatitude() const {  return this->latitude;  }

double Ship::getLongitude() const {  return this->longitude;  }
```

CargoShip.h

```
#ifndef CARGOSHIP_H_
#define CARGOSHIP_H_

#include "Ship.h"

class CargoShip: public Ship {
public:
    CargoShip(const std::string &callSign, double latitude,
              double longitude);
    const std::string getMarker() const;
};

#endif /* CARGOSHIP_H_ */
```

CargoShip.cpp

```
#include "CargoShip.h"

CargoShip::CargoShip(const std::string &callSign, double latitude,
                     double longitude) :
        Ship(callSign, latitude, longitude) {
}

const std::string CargoShip::getMarker() const {
    return "C";
}
```

Ferry.h

```
#ifndef FERRY_H_
#define FERRY_H_

#include "Ship.h"

class Ferry: public Ship {
public:
    Ferry(const std::string &callSign, double latitude,
          double longitude);
    const std::string getMarker() const;
};

#endif /* FERRY_H_ */
```

Ferry.cpp

```
#include "Ferry.h"

Ferry::Ferry(const std::string &callSign, double latitude,
             double longitude) :
        Ship(callSign, latitude, longitude) {
}

const std::string Ferry::getMarker() const {
    return "F";
}
```

ShipFactory.h

```
#ifndef SHIPFACTORY_H_
#define SHIPFACTORY_H_

#include <string>

#include "Ship.h"
#include "CargoShip.h"
#include "Ferry.h"

class ShipFactory {
public:
    static Ship* createShip(const std::string &callSign,
                            double latitude, double longitude);

    static Ferry* createFerry(const std::string &callSign,
                              double latitude, double longitude);

    static CargoShip* createCargoShip(const std::string &callSign,
                              double latitude, double longitude);
};

#endif /* SHIPFACTORY_H_ */
```

ShipFactory.cpp

```
Ship* ShipFactory::createShip(const std::string &callSign,
                              double latitude, double longitude) {
    Ship ship(callSign, latitude, longitude);
    return &ship;
}

Ferry* ShipFactory::createFerry(const std::string &callSign,
                                double latitude, double longitude) {
    Ferry ferry(callSign, latitude, longitude);
    return &ferry;
}

CargoShip* ShipFactory::createCargoShip(const std::string &callSign,
                                double latitude, double longitude) {
    CargoShip cargoShip(callSign, latitude, longitude);
    return &cargoShip;
}
```

---------- Radar.h ----------

```
#ifndef RADAR_H_
#define RADAR_H_

#include <vector>

class Ship;
class Radar {

public:
    /**
     * Returns all ships that have entered the controlled area
     * since the last call to this method.
     */
    std::vector<Ship*> getNewlyEnteredShipsSinceLastPoll();
};

#endif /* RADAR_H_ */
```

---------- Radar.cpp ----------

(ABSICHTLICH WEGGELASSEN)