|  |  |  |
| --- | --- | --- |
| **2019\_2\_java2\_07** | **학번 : 20135151** | **이름 : 이갑성** |

* **내용 점검**

1. 다음 주어진 인터페이스와 클래스를 보고 물음에 답하시오.

|  |
| --- |
| **public** interface Mammal { //Mammal 인터페이스 정의  **void** walk();  void giveBirth();  } |

|  |
| --- |
| **public** **class** Person implements Mammal { //Mammal 인터페이스 구현  **public** **static** String *TYPE* = "사람";  public void walk() { //walk() 재정의  System.*out*.println(*TYPE* + "은 두 발로 걷습니다.");  }  **public** **void** giveBirth() {  System.*out*.println(*TYPE* + "이 아기를 낳습니다.");  }  } |

|  |
| --- |
| **public** **class** Whale implements Mammal { //Mammal 인터페이스 구현  **public** **static** String *TYPE* = "고래";  **public** **void** walk() {  System.*out*.println(*TYPE* + "는 물에서 수영합니다.");  }  Public void giveBirth() { //giveBirth() 재정의  System.*out*.println(*TYPE* + "는 아기를 낳습니다.");  }  } |

1. 위 프로그램에서 빈 부분을 채우시오..
2. 다음 문장이 오류가 발생하는 이유를 설명하시오.

Mammal mal = **new** Mammal();

* 이유: 인터페이스는 자료형으로 선언은 할 수 있어도 객체생성은 불완전하기 때문에 오류발생

1. 다음 소스의 결과를 기술하시오.

Person p = **new** Person();

p.walk();

p.giveBirth();

Whale w = **new** Whale();

w.walk();

w.giveBirth();

* 결과: 사람은 두 발로 걷습니다.

사람이 아기를 낳습니다.

고래는 물에서 수영합니다.

고래는 아기를 낳습니다.

1. 다음 소스의 결과를 기술하시오.

Mammal m[] = {**new** Person(), **new** Whale()};

m[0].walk();

m[0].giveBirth();

m[1].walk();

m[1].giveBirth();

* 결과: 사람은 두 발로 걷습니다.

사람이 아기를 낳습니다.

고래는 물에서 수영합니다.

고래는 아기를 낳습니다.

1. 다음의 인터페이스 선언과 구현에서 잘못된 점을 모두 지적하라.

public interface Edible {

boolean amount;

final int TYPE=10;

public void eat() { }

}

public class Sandwitch extends Edible {

void eat() { }

}

|  |
| --- |
| public interface Edible {  boolean AMOUNT = true;  final int TYPE = 10;  public void eat();  }  public class Sandwitch implements Edible {  public void eat(){  System.out.println(“먹다.”);  }  } |

1. 다음과 같은 조건을 만족하는 프로그램을 작성하라.

(a) 인터페이스 GraphicsObject에서는 int getArea( )과 void draw( )가 선언된다.

(b) GraphicsObject을 구현하는 “Rectangle”클래스를 작성한다.

(c) Rectangle 클래스는 2개의 private 정수 필드인 length와 width을 가진다.

(d) setDimensions (int l, int w) 메소드는 length와 width의 값을 설정한다. 반환값 없음



|  |
| --- |
| **class** Answer{  **public** **static** **void** main(String[] args) {  GraphicsObject obj=**new** Rectangle();  ((Rectangle)obj).setDimensions(34,23);  obj.draw();  }  } |

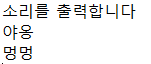
|  |
| --- |
| **[프로그램 소스]**  public interface GraphicsObject {  int getArea();  void draw();  }  public class Rectangle implements GraphicsObject {  private int length;  private int width;    @Override  public int getArea() {  return length \* width;  }    @Override  public void draw() {  System.***out***.println("넓이가 " + getArea() + "사각형을 그립니다.");  }  public void setDimensions(int l, int w) {  this.length = l;  this.width = w;  }  }  public class Answer {  public static void main(String[] args) {  GraphicsObject obj = new Rectangle();  ((Rectangle) obj).setDimensions(34, 23);  obj.draw();  }  } |

* 실행결과 제시

|  |
| --- |
| //타입 변환과 다형성  public interface Vehicle {  public void run();  }  public class Bus implements Vehicle {  @Override  public void run() {  System.out.println("버스가 달립니다.");  }  public void checkFare() {  System.out.println("승차 요금을 체크합니다.");  }  }  public class Taxi implements Vehicle {  @Override  public void run() {  System.out.println("택시가 달립니다.");  }  }  public class Driver {  public void drive(Vehicle vehicle) {  if(vehicle instanceof Bus) {  Bus bus = (Bus) vehicle; //타입 변환  bus.checkFare();  }  vehicle.run();  }  }  public class DriverExample {  public static void main(String[] args) {  Driver driver = new Driver();  Bus bus = new Bus();  Taxi taxi = new Taxi();  driver.drive(bus); //매개변수 다형성  driver.drive(taxi);  }  } |
| **[실행 결과]** |

* 실습 과제

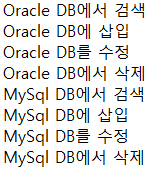
1. 제시된 결과처럼 실행될 수 있도록 Cat과 Dog 클래스를 작성하시오



|  |
| --- |
| **interface** Soundable{  String sound();  }  **//이곳에 Cat과 Dog 클래스 작성 - 본인작성**  **public** **class** Answer1 {  **private** **static** **void** printSound(Soundable sd) {  System.***out***.println(sd.sound());  }  **public** **static** **void** main(String[] args) {  System.***out***.println("소리를 출력합니다");  *printSound*(**new** Cat());  *printSound*(**new** Dog());  }  } |

|  |
| --- |
| **[프로그램 소스]**  interface Soundable {  String sound();  }  public class Dog implements Soundable{    *@Override*  public String sound() {  return "멍멍";  }  }  public class Cat implements Soundable{    *@Override*  public String sound() {  return "야옹";  }  }  public class Answer1 {  private static void printSound(Soundable sd) { //정적메소드  System.***out***.println(sd.sound());  }  public static void main(String[] args) {  System.***out***.println("소리를 출력합니다");  *printSound*(new Cat());  *printSound*(new Dog());  }  } |
| **[실행결과]** |

1. 제시된 실행 결과를 보고 DataAccessObject 인터페이스와 OracleDao와 MySqlDao 구현 클래스를 작성하시오.



|  |
| --- |
| **public** **class** Answer2 {  **public** **static** **void** dbWork(DataAccessObject dao) {  dao.select();  dao.insert();  dao.update();  dao.delete();  }  **public** **static** **void** main(String[] args) {  *dbWork*(**new** OracleDao());  *dbWork*(**new** MySqlDao());  }  } |

|  |
| --- |
| **[프로그램 소스]**  public interface DataAccessObject {  public void select();  public void insert();  public void update();  public void delete();  }  public class OracleDao implements DataAccessObject {  *@Override*  public void select() {  System.***out***.println("Oracle DB에서 검색");  }    *@Override*  public void insert() {  System.***out***.println("Oracle DB에 삽입");  }    *@Override*  public void update() {  System.***out***.println("Oracle DB를 수정");  }    *@Override*  public void delete() {  System.***out***.println("Oracle DB에서 삭제");  }  }  public class MySqlDao implements DataAccessObject{  *@Override*  public void select() {  System.***out***.println("MySql DB에서 검색");  }    *@Override*  public void insert() {  System.***out***.println("MySql DB에 삽입");  }    *@Override*  public void update() {  System.***out***.println("MySql DB를 수정");  }    *@Override*  public void delete() {  System.***out***.println("MySql DB에서 삭제");  }  }  public class Answer2 {  public static void dbWork(DataAccessObject dao) {  dao.select();  dao.insert();  dao.update();  dao.delete();  }  public static void main(String[] args) {  *dbWork*(new OracleDao());  *dbWork*(new MySqlDao());  }  } |
| **[실행결과]** |

1. Action 인터페이스의 익명 구현 객체를 만들어 다음과 같은 실행 결과가 나올 수 있도록 소스를 완성 하시오.



|  |
| --- |
| **interface** Action{  **void** work();  }  **public** **class** Answer {  **public** **static** **void** main(String[] args) {  Action action= //이곳에 익명 클래스 구현 - 본인작성  action.work();  }  } |

|  |
| --- |
| **[프로그램 소스]**  public interface Action {    void work();  }  public class Answer {  public static void main(String[] args) {  Action action = new Action() {  public void work() {  System.***out***.println("무명클래스로 인터페이스를 구현");  System.***out***.println("무명 객체로 work() 메소드를 호출합니다.");  }  };    action.work();  }  } |
| **[실행결과]** |

1. Person이라는 클래스를 정의하라. Person은 이름(name)과 키(height)를 필드로 가진다. Person은 Comparable 인터페이스를 구현한다. 이 Comparable 인터페이스를 이용하여서 가장 키 큰 사람의 이름을 반환하는 메소드 구현하고 테스트하라.

* Person 클래스 정의

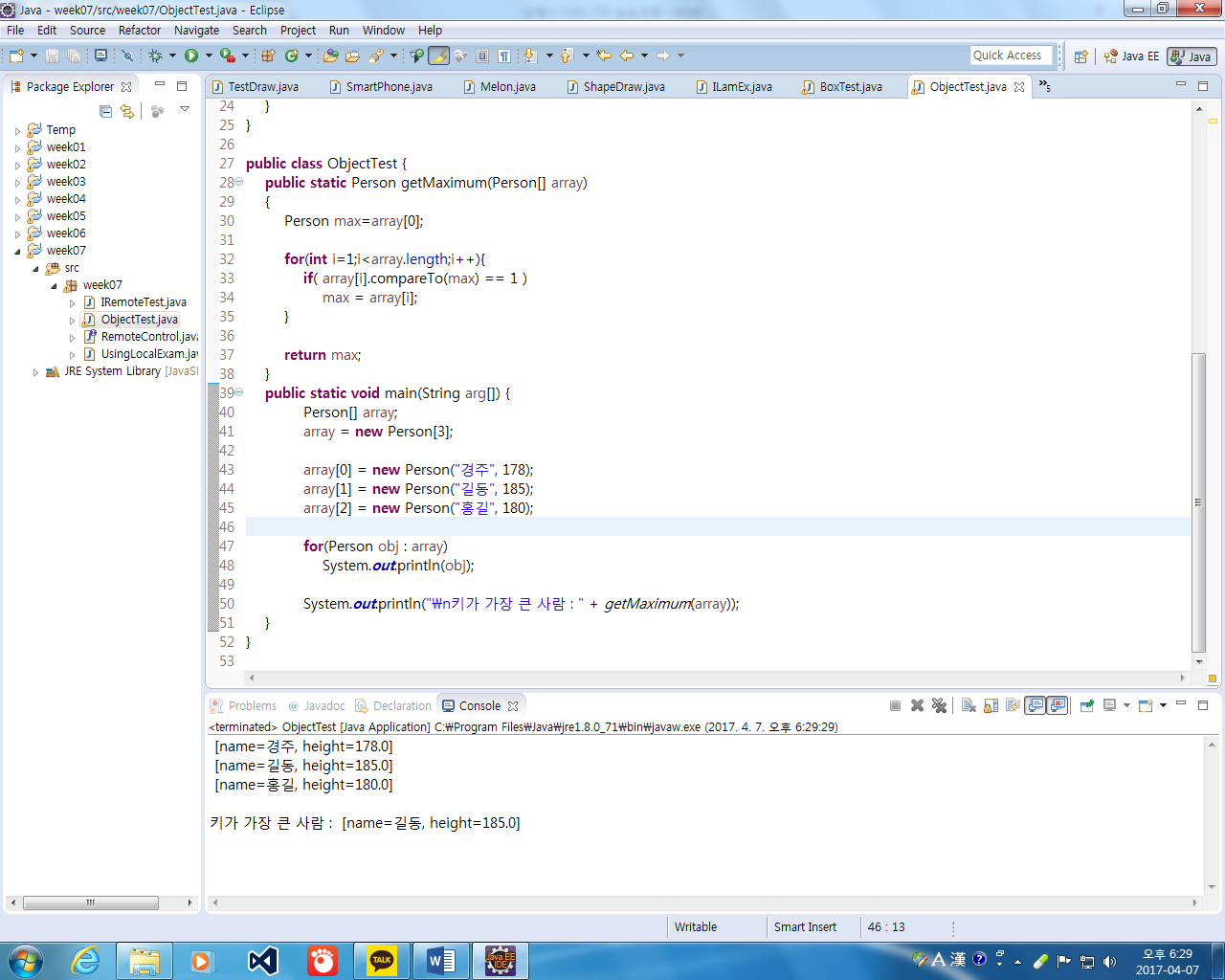
- compareTo() 메소드 재정의 : Person 객체의 height를 비교

- 객체 내용을 문자열로 반환 하는 toString()

* ObjectTest 클래스 정의

- getMaximum(Person[] array) : 가장 키가 큰 사람 반환(Person), compareTo() 메소드 사용, 정적메소드

- main() : Person 객체 배열을 사용하여 5명의 이름과 키 초기화, 키가 가장 큰 사람 출력



|  |
| --- |
| **[프로그램 소스]**    public interface IComparable {  public String getHighest(Person[] array);  }  public class Person implements IComparable {  private String name;  private double height;  public Person(String name, double height) {  this.name = name;  this.height = height;  }  public String getName() {  return name;  }  public double getHeight() {  return height;  }  *@Override*  public String getHighest(Person[] array) {  Person highest = array[0];  String highName = "";  for (int i = 1; i < array.length; i++) {  if (highest.compareTo(array[i]) == -1) {  highest = array[i];  highName = array[i].getName();  }  }  return highName;  }  public int compareTo(Object per) {  Person p = null;  if (per instanceof Person) {  p = (Person) per;  }  if (this.getHeight() < p.getHeight()) {  return -1;  } else if (this.getHeight() > p.getHeight()) {  return 1;  } else  return 0;  }  public String toString() {  String result = "[name = " + name + ", height = " + height + "]";  return result;  }  }  public class ObjectTest{  public static void main(String[] args) {  Person[] pArray = new Person[5];  pArray[0] = new Person("스펀지밥", 50.5);  pArray[1] = new Person("뚱이", 100.1);  pArray[2] = new Person("징징이", 120.8);  pArray[3] = new Person("짱구", 150.1);  pArray[4] = new Person("짱아", 100.3);    for (int i = 0; i < pArray.length; i++) {  System.***out***.println(pArray[i]);  }  System.***out***.println();  System.***out***.println("키가 가장 큰 사람: " + *getMaximum*(pArray));  }    public static Person getMaximum(Person[] array) {  Person max = array[0];    for(int i = 1; i < array.length; i++) {  if(max.compareTo(array[i]) == -1) {  max = array[i];  }  }    return max;  }    } |
| **[실행 결과]** |