**package** homework2;

**public** **class** SolutionTo1 {

//methods of an interface are abstract by default and cannot be implemented. Versus an abstract class can have a method that implements a default behavior

}

Problem 2 output:

**package** homework2;

**public** **class** Rodent {

**public** **void** location(){};

**public** **void** tailtype(){};

**public** **void** lifespan(){};

}

**class** Mouse **extends** Rodent{

**public** **void** location(){

System.*out*.println("Everywhere");

}

**public** **void** tailtype(){

System.*out*.println("No hair");

}

**public** **void** lifespan(){

System.*out*.println("1-3 years");

}

}

**class** Gerbil **extends** Rodent{

**public** **void** location(){

System.*out*.println("Africa and Asia and your local petsmart");

}

**public** **void** tailtype(){

System.*out*.println("long");

}

**public** **void** lifespan(){

System.*out*.println("2-3 years");

}

}

**class** Hamster **extends** Rodent{

**public** **void** location(){

System.*out*.println("Europe and Asia and your local petsmart");

}

**public** **void** tailtype(){

System.*out*.println("short");

}

**public** **void** lifespan(){

System.*out*.println("3-4 years if lucky");

}

}

**class** Rodents{

**public** **static** **void** main(String[] args){

Rodent r[] = **new** Rodent[6];

**for**(**int** i=0; i<r.length; i++){

r[i]=*randomAnimal*();

}

**for**(**int** i=0; i<r.length; i++){

r[i].location();

r[i].tailtype();

r[i].lifespan();

}

}

**public** **static** Rodent randomAnimal(){

**switch**((**int**)(Math.*random*()\*4)){

**default**:

**case** 0: **return** **new** Gerbil();

**case** 1: **return** **new** Mouse();

**case** 2: **return** **new** Hamster();

}

}

}

Problem 2 output:

Africa and Asia and your local petsmart

long

2-3 years

Africa and Asia and your local petsmart

long

2-3 years

Europe and Asia and your local petsmart

short

3-4 years if lucky

Africa and Asia and your local petsmart

long

2-3 years

Everywhere

No hair

1-3 years

Africa and Asia and your local petsmart

long

2-3 years

Problem 3

**package** homework2;

**abstract** **class** One {

**public** **static** **void** nonAbstract() {

System.*out*.print("Abstract class test");

}

}

**class** poly {

**public** **static** **void** main(String []args) {

One two = **new** One() {};

One.*nonAbstract*();

}

}

//Create a class as abstract without including any abstract methods, and verify that you cannot create an instance of that class. This code proves that an instance cannot be created based on the rules and compilation

Problem 5

**package** homework2;

**public** **class** Problem5 {

**static** **interface** Instrument {

**void** play(String notes);

**void** adjust();

}

**static** **class** Wind **implements** Instrument {

**public** **void** play(String notes) {

System.*out*.println(notes + "Wind.play()");

}

**public** **void** adjust() { System.*out*.println(notes + ".adjust() "); }

}

**static** **class** Percussion **implements** Instrument {

**public** **void** play(String notes) {

System.*out*.println(notes + "Percussion.play() ");

}

**public** **void** adjust() { System.*out*.println(notes + ".adjust() "); }

}

**static** **class** Brass **extends** Wind {

}

**static** **class** Woodwind **extends** Wind {

}

**public** **static** **class** Music {

**public** **static** **void** main(String[] args) {

Brass brass = **new** Brass();

Woodwind woody = **new** Woodwind();

Percussion percus = **new** Percussion();

brass.play("notes");

woody.play("notes");

percus.play("notes");

}

}

}

//output

//notesWind.play()

//notesWind.play()

//notesPercussion.play()

Abstract Rodent

**package** homework2;

**public** **abstract** **class** RodentAbstract {

**public** **void** location(){};

**public** **void** tailtype(){};

**public** **void** lifespan(){};

}