/\*\*

\* Abstract Pokemon class has type, status, and health and

\* can fight other Pokemon

\* **@author** Hunter Damron

\* //Pokemon.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**import** java.util.Random;

**public** **abstract** **class** **Pokemon** {

**protected** **String** name, species;

**protected** *Attack* type;

**protected** *Damage* status;

**protected** **int** health, statusCounter;

**private** **Random** gen = **new** Random();

**public** **static** **final** **int** ***ATTACK\_MISS*** = 0,

***ATTACK\_SUCCESS*** = 1, ***STATUS\_SUCCESS*** = 2;

/\*\*

\* Attack method implemented in subclass

\* **@param** opposition Pokemon recieving the attack

\* **@return** Returns int reflecting success

\*/

**public** **abstract** **int** **attack**(**Pokemon** opposition);

/\*\*

\* Hurts the Pokemon

\* **@param** damage Damage inflicted if successful

\* **@param** accuracy Accuracy of attack (of 100)

\* **@return** Returns success

\*/

**public** **boolean** **hurt**(**int** damage, **int** accuracy) {

**if**(gen.nextInt(100) < accuracy) {

health -= damage;

**return** **true**;

} **else** **return** **false**;

}

/\*\*

\* Inflicts the pokemon with a status

\* **@param** infliction Status to be inflicted on pokemon

\* **@param** strength Length of time that the pokemon is inflicted

\* **@param** chance Change that the poison will be successful (of 10)

\* **@return** Returns success

\*/

**public** **boolean** **inflictStatus**(*Damage* infliction, **int** strength, **int** chance) {

**if**(gen.nextInt(10) < chance) {

status = infliction;

statusCounter = strength;

**return** **true**;

} **else** **return** **false**;

}

/\*\*

\* Tests if the pokemon is dead

\* **@return** Returns true if the pokemon is dead

\*/

**public** **boolean** **isDead**() {

**if**(health <= 0) {

status = *Damage*.***DEAD***;

**return** **true**;

} **else** **return** **false**;

}

/\*\*

\* Inflicts damage from pending afflictions and

\* counts down the time remaining on the affliction

\*/

**public** **void** **inflictDamage**() {

health -= status.damage;

statusCounter--;

**if**(statusCounter == 0)

status = *Damage*.***HEALTHY***;

}

/\*\*

\* Gets the name of the pokemon

\* **@return** Returns the name of the pokemon

\*/

**public** **String** **getName**() {

**return** name;

}

/\*\*

\* Gets the status afflicting the pokemon

\* **@return** Returns the pokemon's status

\*/

**public** *Damage* **getStatus**() {

**return** status;

}

/\*\*

\* Overrides Object.toString() method to print the pokemon's

\* name, species, health, type, and status (if applicable)

\* **@return** Returns the pokemon represented as a string

\*/

**public** **String** **toString**() {

**String** **message** = "" + species + " named " + name + " [Health = " + health + ", “

+ “Type = " + type;

**if**(status != *Damage*.***HEALTHY***)

message += ", Status = " + status;

message += "]";

**return** message;

}

}

/\*\*

\* Extends Pokemon - Fire type Pokemon capable of causing burns

\* **@author** Hunter Damron

\* //Charmeleon.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**public** **class** **Charmeleon** **extends** **Pokemon** {

**public** **Charmeleon**(**String** name) {

**this**.name = name;

type = *Attack*.***FIRE***;

status = *Damage*.***HEALTHY***;

health = 100;

species = "Charmeleon";

}

/\*\*

\* Overrides Pokemon attack method (does 20 damage and can burn)

\* **@return** Returns value representing success of attack

\*/

**public** **int** **attack**(**Pokemon** opposition) {

**if**(status.canAttack && opposition.hurt(20, 85)) {

**if**(opposition.inflictStatus(*Damage*.***BURN***, 2, 3))

**return** **Pokemon**.***STATUS\_SUCCESS***;

**else** **return** **Pokemon**.***ATTACK\_SUCCESS***;

} **else** **return** **Pokemon**.***ATTACK\_MISS***;

}

}

/\*\*

\* Extends Pokemon - Bug type Pokemon capable of poisoning

\* **@author** Hunter Damron

\* //Seviper.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**public** **class** **Seviper** **extends** **Pokemon** {

**public** **Seviper**(**String** name) {

**this**.name = name;

type = *Attack*.***BUG***;

status = *Damage*.***HEALTHY***;

health = 140;

species = "Seviper";

}

/\*\*

\* Overrides Pokemon attack method (does 10 damage and can poison)

\* **@return** Returns value representing success of attack

\*/

**public** **int** **attack**(**Pokemon** opposition) {

**if**(status.canAttack && opposition.hurt(10, 95)) {

**if**(opposition.inflictStatus(*Damage*.***POISON***, 4, 3))

**return** **Pokemon**.***STATUS\_SUCCESS***;

**else** **return** **Pokemon**.***ATTACK\_SUCCESS***;

} **else** **return** **Pokemon**.***ATTACK\_MISS***;

}

}

/\*\*

\* Tests the polymorphism of classes which extend Pokemon

\* **@author** Hunter Damron

\* //PokemonPolymorphismTest.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**import** java.util.Arrays;

**public** **class** **PokemonPolymorphismTest** {

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

**Pokemon**[] **poke** = **new** **Pokemon**[] {**new** Seviper("Fang"),

**new** Charmeleon("Supa Hot Fire")};

**int** **success**;

**System**.***out***.println("Expected Output: [Seviper named Fang "

+ "[Health = 140, Type = Bug], Charmeleon named Supa "

+ "Hot Fire [Health = 100, Type = Fire]]");

**System**.***out***.println("Actual Ouput: " + **Arrays**.*deepToString*(poke) + "\n");

success = poke[0].attack(poke[1]);

**System**.***out***.println("Fang's turn to attack:\n----------------------");

**if**(success == **Pokemon**.***STATUS\_SUCCESS***) {

**System**.***out***.println("Expected Output: \tCharmeleon named Supa Hot "

+ "Fire [Health = 90, Type = Fire, Status = Poisoned]");

**System**.***out***.println("Actual Output: \t" + poke[1]);

**System**.***out***.println("\*Damage from poison is then inflicted");

poke[1].inflictDamage();

**System**.***out***.println("Expected Output: \tCharmeleon named Supa Hot "

+ "Fire [Health = 85, Type = Fire, Status = Poisoned]");

**System**.***out***.println("Actual Output: \t" + poke[1] + "\n");

} **else** **if**(success == **Pokemon**.***ATTACK\_SUCCESS***) {

**System**.***out***.println("Expected Output: \tCharmeleon named Super Hot "

+ "Fire [Health = 90, Type = Fire]");

**System**.***out***.println("Actual Output: \t" + poke[1] + "\n");

poke[1].inflictDamage();

} **else** {

**System**.***out***.println("No change. Attack missed\n");

}

**System**.***out***.println("Supa Hot Fire's turn to attack:"

+ "\n--------------------------------");

success = poke[1].attack(poke[0]);

**if**(success == **Pokemon**.***STATUS\_SUCCESS***) {

**System**.***out***.println("Expected Output: \tSeviper named Fang [Health = 120, "

+ "Type = Bug, Status = Burnt]");

**System**.***out***.println("Actual Output: \t" + poke[0]);

**System**.***out***.println("\*Damage from burn is then inflicted");

poke[0].inflictDamage();

**System**.***out***.println("Expected Output: \tSeviper named Fang [Health = 110, "

+ "Type = Bug, Status = Burnt]");

**System**.***out***.println("Actual Output: \t" + poke[0]);

} **else** **if**(success == **Pokemon**.***ATTACK\_SUCCESS***) {

**System**.***out***.println("Expected Output: \tSeviper named Fang [Health = 120, "

+ "Type = Bug]");

**System**.***out***.println("Actual Output: \t" + poke[0] + "\n");

poke[1].inflictDamage();

} **else** {

**System**.***out***.println("No change. Attack missed\n");

}

}

}

/\*\*

\* Provides attack types for Pokemon (not yet necessary for basic version)

\* **@author** Hunter Damron

\* //Attack.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**public** **enum** *Attack* {

***NORMAL***("Normal"), ***FIRE***("Fire"), ***WATER***("Water"), ***GRASS***("Grass"),

***ELECTRIC***("Electric"), ***ICE***("Ice"), ***FIGHTING***("Fighting"), ***BUG***("Bug"),

***PSYCHIC***("Psychic"), ***GHOST***("Ghost"), ***GROUND***("Ground");

**public** **String** name;

/\*\*

\* Constructor associates enumeration with name

\* **@param** name

\*/

**private** **Attack**(**String** name) {

**this**.name = name;

}

/\*\*

\* Gets the attack type's name

\* **@return** Returns name

\*/

**public** **String** **toString**() {

**return** name;

}

}

/\*\*

\* Provides damage from from special statuses that can affect pokemon

\* **@author** Hunter Damron

\* //Damage.java

\* //Honor Code: I did not lie, cheat, or steal

\*/

**public** **enum** *Damage* {

***HEALTHY***(0, **true**, "Healthy"), ***BURN***(10, **true**, "Burnt"), ***POISON***(5, **true**, "Poisoned"),

***CONFUSION***(0, **false**, "Confused"), ***SLEEP***(0, **false**, "Asleep"), ***FREEZE***(0, **false**, "Frozen"),

***PARALYSIS***(0, **false**, "Paralyzed"), ***DEAD***(0, **false**, "Dead");

**public** **int** damage;

**public** **boolean** canAttack;

**public** **String** name;

/\*\*

\* Constructor associates enumeration with its default variables

\* **@param** damage Damage done per turn by status

\* **@param** canAttack Determines if the pokemon can attack

\* **@param** name Name of affliction

\*/

**private** **Damage**(**int** damage, **boolean** canAttack, **String** name) {

**this**.damage = damage;

**this**.canAttack = canAttack;

**this**.name = name;

}

/\*\*

\* Gets the status type's name

\* **@return** Returns name

\*/

**public** **String** **toString**() {

**return** name;

}

}

/\*\*\*\*\*\*\*\*\*\*/

/\* Output \*/

/\*\*\*\*\*\*\*\*\*\*/

##Case 1 – Attack and Affliction both hit##

-------------------------------------------

Expected Output: [Seviper named Fang [Health = 140, Type = Bug], Charmeleon named Supa Hot Fire [Health = 100, Type = Fire]]

Actual Ouput: [Seviper named Fang [Health = 140, Type = Bug], Charmeleon named Supa Hot Fire [Health = 100, Type = Fire]]

Fang's turn to attack:

----------------------

Expected Output: Charmeleon named Supa Hot Fire [Health = 90, Type = Fire, Status = Poisoned]

Actual Output: Charmeleon named Supa Hot Fire [Health = 90, Type = Fire, Status = Poisoned]

\*Damage from poison is then inflicted

Expected Output: Charmeleon named Supa Hot Fire [Health = 85, Type = Fire, Status = Poisoned]

Actual Output: Charmeleon named Supa Hot Fire [Health = 85, Type = Fire, Status = Poisoned]

Supa Hot Fire's turn to attack:

--------------------------------

Expected Output: Seviper named Fang [Health = 120, Type = Bug, Status = Burnt]

Actual Output: Seviper named Fang [Health = 120, Type = Bug, Status = Burnt]

\*Damage from burn is then inflicted

Expected Output: Seviper named Fang [Health = 110, Type = Bug, Status = Burnt]

Actual Output: Seviper named Fang [Health = 110, Type = Bug, Status = Burnt]

##Case 2 – Attack hits but Affliction misses##

----------------------------------------------

Expected Output: [Seviper named Fang [Health = 140, Type = Bug], Charmeleon named Supa Hot Fire [Health = 100, Type = Fire]]

Actual Ouput: [Seviper named Fang [Health = 140, Type = Bug], Charmeleon named Supa Hot Fire [Health = 100, Type = Fire]]

Fang's turn to attack:

----------------------

Expected Output: Charmeleon named Super Hot Fire [Health = 90, Type = Fire]

Actual Output: Charmeleon named Supa Hot Fire [Health = 90, Type = Fire]

Supa Hot Fire's turn to attack:

--------------------------------

Expected Output: Seviper named Fang [Health = 120, Type = Bug]

Actual Output: Seviper named Fang [Health = 120, Type = Bug]

##Case 3 – Attack misses##

--------------------------

Expected Output: [Seviper named Fang [Health = 140, Type = Bug], Charmeleon named Supa Hot Fire [Health = 100, Type = Fire]]

Actual Ouput: [Seviper named Fang [Health = 140, Type = Bug], Charmeleon named Supa Hot Fire [Health = 100, Type = Fire]]

Fang's turn to attack:

----------------------

Expected Output: Charmeleon named Super Hot Fire [Health = 90, Type = Fire]

Actual Output: Charmeleon named Supa Hot Fire [Health = 90, Type = Fire]

Supa Hot Fire's turn to attack:

--------------------------------

No change. Attack missed

