

## INDICE

• Pokemon . . . . .	1-2
• PokemonTest . . . . .	3
○ Agua . . . . .	4
○ AguaTest . . . . .	5
○ Fuego . . . . .	6
○ FuegoTest . . . . .	7
○ Planta . . . . .	8
○ PlantaTest . . . . .	9
• ListaPokemon . . . . .	10
• ListaPokemonTest . . . . .	11
• ListaPokemonAElegir . . . . .	12
• Jugador . . . . .	13
• JugadorTest . . . . .	14
• ListaJugador . . . . .	15-16
• ListaJugadorTest . . . . .	17
• Ataque . . . . .	18
• AtaqueTest . . . . .	19
• ListaAtaque . . . . .	20
• ListaAtaqueTest . . . . .	21
• Teclado . . . . .	22
• Tablero . . . . .	23-25
• Exceptions . . . . .	26

```

1 package org.pmoor.packproyecto;
2
3 public abstract class Pokemon {
4     private String nombre;
5     private String tipo;
6     private int vida;
7     private ListaAtaque listaAtaques=new ListaAtaque();
8
9     public Pokemon(String pNombre, String pTipo, int pVida) {
10         this.nombre=pNombre;
11         this.tipo=pTipo;
12         this.vida=pVida;
13         this.listaAtaques=crearListaAtaques();
14     }
15
16     public void imprimir() {
17         System.out.println("\nNOMBRE : " + this.nombre + ", VIDA : " + this.vida + ", TIPO : " + this.tipo);
18         this.listaAtaques.imprimir();
19     }
20
21     private int usarAtaque(int pPosAtaqueEnLista) {
22         return this.listaAtaques.usarAtaque(pPosAtaqueEnLista);
23     }
24
25     public void cambiarVida(int pDaño) {
26         this.vida=this.vida-pDaño;
27     }
28
29     public boolean muerto() {
30         boolean muerto=false;
31         if(this.vida<=0) {
32             muerto=true;
33         }
34         return muerto;
35     }
36
37     public void atacar(Pokemon pPK) throws CambiarDePokemonException, UsarCuraException{
38         int num=0;
39         boolean repetir=true;
40         int daño=0;
41         while(num==0||repetir) {
42             num=Teclado.getTeclado().leerNumero(6);
43             if(num==5) {
44                 throw(new CambiarDePokemonException());
45             }
46             else if(num==6) {
47                 throw(new UsarCuraException());
48             }
49             daño=this.usarAtaque(num-1);
50             if((num<1&&num>4)||daño== -100000) {
51                 System.out.println("ERROR: Su respuesta no es la esperada.");
52                 System.out.println("¡VUELVE A INTENTARLO!");
53             }
54         }
55     }

```

```
54     else{
55         repetir=false;
56     }
57 }
58 if(this.supereficaz(pPK)) {
59     daño=daño+(daño/5);
60     System.out.println("Tu ataque es SUPEREFICAZ");
61 }
62 else if(this.pocoeficaz(pPK)) {
63     daño=daño-(daño/5);
64     System.out.println("Tu ataque es POCO EFICAZ");
65 }
66 pPK.cambiarVida(daño);
67 }
68
69 protected abstract ListaAtaque crearListaAtaques();
70
71 protected abstract boolean supereficaz(Pokemon pPK);
72
73 protected abstract boolean pocoeficaz(Pokemon pPK);
74
75 public void imprimirSoloInfoPK() {
76     System.out.println("NOMBRE : " + this.nombre + ", VIDA : " + this.vida +", TIPO : " + this.tipo);
77 }
78
79 public int getVida() {
80     return this.vida;
81 }
82 }
```

```
1 package org.pmoa.packproyecto;
2
3+ import static org.junit.Assert.*;
4
5
6 public class PokemonTest {
7
8
9     private Agua pk1;
10    private Fuego pk2;
11    private Planta pk3;
12
13    @Before
14    public void setUp() throws Exception {
15        pk1 = new Agua("Pk1", "Agua", 10);
16        pk2 = new Fuego("Pk2", "Fuego", 3);
17        pk3 = new Planta("Pk3", "Planta", 11);
18    }
19
20
21    @After
22    public void tearDown() throws Exception {
23    }
24
25    @Test
26    public void testPokemon() {
27        assertNotNull(pk1);
28    }
29
30    @Test
31    public void testCambiarVida() {
32        pk1.cambiarVida(5);
33        assertFalse(pk1.muerto());
34    }
35
36    @Test
37    public void testMuerto() {
38        pk1.cambiarVida(10);
39        assertTrue(pk1.muerto());
40    }
41 }
```

```
1 package org.pmoo.packproyecto;
2
3 public class Agua extends Pokemon{
4     public Agua(String pNombre, String pTipo, int pVida) {
5         super(pNombre, pTipo, pVida);
6     }
7
8     protected ListaAtaque crearListaAtaques() {
9         ListaAtaque lista=new ListaAtaque();
10        Ataque atq1=new Ataque("Tsunami", "Agua", 25, 3);
11        lista.añadirAtaque(atq1);
12        Ataque atq2=new Ataque("Manguerazo", "Agua", 5, 20);
13        lista.añadirAtaque(atq2);
14        Ataque atq3=new Ataque("Remolino", "Agua", 10, 15);
15        lista.añadirAtaque(atq3);
16        Ataque atq4=new Ataque("Maremoto", "Agua", 20, 8);
17        lista.añadirAtaque(atq4);
18        return lista;
19    }
20
21    @Override
22    protected boolean supereficaz(Pokemon pPK) {
23        boolean supereficaz=false;
24        if(pPK instanceof Fuego) {
25            supereficaz=true;
26        }
27        return supereficaz;
28    }
29
30    @Override
31    protected boolean pocoeficaz(Pokemon pPK) {
32        boolean pocoeficaz=false;
33        if(pPK instanceof Planta) {
34            pocoeficaz=true;
35        }
36        return pocoeficaz;
37    }
38 }
```

```
1 package org.pmoa.packproyecto;
2
3+import static org.junit.Assert.*;□
4
5 public class AguaTest {
6     Agua p1;
7     Fuego p2;
8     Planta p3;
9
10
11    @Before
12    public void setUp() throws Exception {
13        p1= new Agua("Squirtle","Agua",100);
14        p2= new Fuego ("Charmander","Fuego",100);
15        p3= new Planta ("Bulbasur","Planta",100);
16    }
17
18    @After
19    public void tearDown() throws Exception {
20    }
21
22    @Test
23    public void testAgua() {
24        assertNotNull(p1);
25    }
26
27    @Test
28    public void superEficazTest() {
29        assertTrue(p1.supereficaz(p2));
30        assertFalse(p1.supereficaz(p3));
31    }
32
33    @Test
34    public void pocoEficazTest() {
35        assertFalse(p1.pocoeficaz(p2));
36        assertTrue(p1.pocoeficaz(p3));
37    }
38
39    }
40
41 }
```

```
1 package org.pmoo.packproyecto;
2
3 public class Fuego extends Pokemon{
4     public Fuego(String pNombre, String pTipo, int pVida) {
5         super(pNombre, pTipo, pVida);
6     }
7
8     protected ListaAtaque crearListaAtaques() {
9         ListaAtaque lista=new ListaAtaque();
10        Ataque atq1=new Ataque("Ascuas", "Fuego", 10, 15);
11        lista.añadirAtaque(atq1);
12        Ataque atq2=new Ataque("Llamarada", "Fuego", 20, 8);
13        lista.añadirAtaque(atq2);
14        Ataque atq3=new Ataque("Bomba termica", "Fuego", 25, 3);
15        lista.añadirAtaque(atq3);
16        Ataque atq4=new Ataque("Antibomberos", "Fuego", 5, 20);
17        lista.añadirAtaque(atq4);
18        return lista;
19    }
20
21    @Override
22    protected boolean supereficaz(Pokemon pPK) {
23        boolean supereficaz=false;
24        if(pPK instanceof Planta) {
25            supereficaz=true;
26        }
27        return supereficaz;
28    }
29
30    @Override
31    protected boolean pocoeficaz(Pokemon pPK) {
32        boolean pocoeficaz=false;
33        if(pPK instanceof Agua) {
34            pocoeficaz=true;
35        }
36        return pocoeficaz;
37    }
38}
```

```
1 package org.pmoa.packproyecto;
2
3+import static org.junit.Assert.*;
4
5
6 public class FuegoTest {
7     Fuego p1;
8     Agua p2;
9     Planta p3;
10
11
12
13
14@Before
15 public void setUp() throws Exception {
16     p1= new Fuego ("Charmander","Fuego",100);
17     p2= new Agua("Squirtle","Agua",100);
18     p3 = new Planta ("Bulbasur","Planta",100);
19 }
20
21@After
22 public void tearDown() throws Exception {
23 }
24
25@Test
26 public void testFuego() {
27     assertNotNull(p1);
28 }
29
30@Test
31 public void superEficazTest() {
32     assertTrue(p1.supereficaz(p3));
33     assertFalse(p2.supereficaz(p2));
34 }
35
36@Test
37 public void pocoEficazTest() {
38     assertFalse(p1.pocoeficaz(p3));
39     assertTrue(p1.pocoeficaz(p2));
40 }
41 }
```

```
1 package org.pmoo.packproyecto;
2
3 public class Planta extends Pokemon{
4     public Planta(String pNombre, String pTipo, int pVida) {
5         super(pNombre, pTipo, pVida);
6     }
7
8     protected ListaAtaque crearListaAtaques() {
9         ListaAtaque lista=new ListaAtaque();
10        Ataque atq1=new Ataque("Ortiga Salvaje", "Planta", 20, 8);
11        lista.añadirAtaque(atq1);
12        Ataque atq2=new Ataque("Fotosintesis", "Planta", 10, 15);
13        lista.añadirAtaque(atq2);
14        Ataque atq3=new Ataque("Espora", "Planta", 5, 20);
15        lista.añadirAtaque(atq3);
16        Ataque atq4=new Ataque("Abrazo de Cactus", "Planta", 25, 3);
17        lista.añadirAtaque(atq4);
18        return lista;
19    }
20
21    @Override
22    protected boolean supereficaz(Pokemon pPK) {
23        boolean supereficaz=false;
24        if(pPK instanceof Agua) {
25            supereficaz=true;
26        }
27        return supereficaz;
28    }
29
30    @Override
31    protected boolean pocoeficaz(Pokemon pPK) {
32        boolean pocoeficaz=false;
33        if(pPK instanceof Fuego) {
34            pocoeficaz=true;
35        }
36        return pocoeficaz;
37    }
38 }
```

```
1 package org.pmoa.packproyecto;
2
3+import static org.junit.Assert.*;
4
5 public class PlantaTest {
6     Planta p1;
7     Fuego p2;
8     Agua p3;
9
10
11    @Before
12    public void setUp() throws Exception {
13        p1 = new Planta ("Bulbasur","Planta",100);
14        p2=new Fuego ("Charmander","Fuego",100);
15        p3=new Agua("Squirtle","Agua",100);
16    }
17
18    @After
19    public void tearDown() throws Exception {
20    }
21
22
23    @Test
24    public void testPlanta() {
25        assertNotNull(p1);
26    }
27
28
29    @Test
30    public void superEficazTest() {
31        assertTrue(p1.supereficaz(p3));
32        assertFalse(p1.supereficaz(p2));
33    }
34
35
36    @Test
37    public void pocoEficazTest() {
38        assertFalse(p1.pocoeficaz(p3));
39        assertTrue(p1.pocoeficaz(p2));
40    }
41 }
```

```
1 package org.pmoo.packproyecto;
2
3+ import java.util.ArrayList;[]
4
5
6 public class ListaPokemon {
7     private ArrayList<Pokemon> listaPokemon;
8
9+     public ListaPokemon() {
10         this.listaPokemon=new ArrayList<Pokemon>();
11     }
12
13+     private Iterator<Pokemon> getIterador(){
14         return this.listaPokemon.iterator();
15     }
16
17+     public void imprimir() {
18         Iterator<Pokemon> itr=this.getIterador();
19         Pokemon pokemon;
20         int pos=1;
21         while(itr.hasNext()) {
22             pokemon=itr.next();
23             System.out.println("\n"+pos+"."+" ");
24             pokemon.imprimir();
25             |
26             pos++;
27         }
28     }
29
30+     public Pokemon sacarPokemon(int pPosPK) throws Exception{
31         Pokemon pk=null;
32         pk=this.listaPokemon.get(pPosPK);
33         return pk;
34     }
35
36+     public boolean listaVacia() {
37         boolean vacia=false;
38         if(this.listaPokemon.size()==0) {
39             vacia=true;
40         }
41         return vacia;
42     }
43
44+     public void eliminarPokemon(Pokemon pPK) {
45         this.listaPokemon.remove(pPK);
46     }
47
48+     public void añadirPokemon(Pokemon pPK) {
49         this.listaPokemon.add(pPK);
50     }
51 }
```

```
3+ import static org.junit.Assert.*;□
8
9 public class ListaPokemonTest {
10     private Agua pk1;
11     private Fuego pk2;
12     private Planta pk3;
13     private ListaPokemon lp;
14     private ListaPokemon lp2;
15@    @Before
16     public void setUp() throws Exception {
17         lp = new ListaPokemon();
18         lp2 = new ListaPokemon();
19         pk1 = new Agua("Pk1", "Agua", 10);
20         pk2 = new Fuego("Pk2", "Fuego", 3);
21         pk3 = new Planta("Pk3", "Planta", 11);
22         lp.añadirPokemon(pk1);
23         lp.añadirPokemon(pk2);
24         lp.añadirPokemon(pk3);
25     }
26
27@    @After
28     public void tearDown() throws Exception {
29     }
30
31@    @Test
32     public void testListaPokemon() {
33         assertNotNull(lp);
34     }
35
36@    @Test
37     public void testSacarPokemon() {
38         try {
39             Pokemon p = lp.sacarPokemon(3);
40             assertEquals(p,pk1);
41         } catch (Exception e) {}
42     }
43
44@    @Test
45     public void testListaVacia() {
46         assertFalse(lp.listaVacia());
47         assertTrue(lp2.listaVacia());
48     }
49
50@    @Test
51     public void testEliminarPokemon() {
52         lp.eliminarPokemon(pk2);
53         Pokemon p;
54         try {
55             p = lp.sacarPokemon(4);
56             assertNotEquals(p,pk2);
57         } catch (Exception e) {}
58     }
59
60@    @Test
61     public void testAñadirPokemon() □
62         Agua pk4 = new Agua("PK4", "Agua", 8);
63         lp.añadirPokemon(pk4);
64         Pokemon p;
65         try {
66             p = lp.sacarPokemon(6);
67             assertEquals(p,pk4);
68         } catch (Exception e) {}
69     }
70 }
```

```
1 package org.pmoo.packproyecto;
2
3 public class ListaPokemonAElegir {
4
5     private ListaPokemon listaPAE;
6     private static ListaPokemonAElegir miLPAE=new ListaPokemonAElegir();
7
8     private ListaPokemonAElegir() {
9         this.listaPAE = new ListaPokemon();
10        this.crearListaPokemons();
11    }
12
13     public static ListaPokemonAElegir getListPokemonAElegir() {
14         return miLPAE;
15     }
16
17     private void crearListaPokemons() {
18         Fuego Pokemon1 = new Fuego ("Typhlosion", "Fuego", 110);
19         this.listaPAE.añadirPokemon(Pokemon1);
20         Fuego Pokemon2 = new Fuego ("Infernia", "Fuego", 110);
21         this.listaPAE.añadirPokemon(Pokemon2);
22         Planta Pokemon3 = new Planta ("Venusaur", "Planta", 90);
23         this.listaPAE.añadirPokemon(Pokemon3);
24         Planta Pokemon4 = new Planta ("Sylvanis", "Planta", 90);
25         this.listaPAE.añadirPokemon(Pokemon4);
26         Agua Pokemon5 = new Agua ("Greninja", "Agua", 100);
27         this.listaPAE.añadirPokemon(Pokemon5);
28         Agua Pokemon6 = new Agua ("Marinna", "Agua", 100);
29         this.listaPAE.añadirPokemon(Pokemon6);
30         Fuego Pokemon7 = new Fuego ("Charizard", "Fuego", 100);
31         this.listaPAE.añadirPokemon(Pokemon7);
32         Fuego Pokemon8 = new Fuego ("Flamaris", "Fuego", 100);
33         this.listaPAE.añadirPokemon(Pokemon8);
34         Planta Pokemon9 = new Planta ("Sceptile", "Planta", 110);
35         this.listaPAE.añadirPokemon(Pokemon9);
36         Planta Pokemon10 = new Planta ("Petalina", "Planta", 110);
37         this.listaPAE.añadirPokemon(Pokemon10);
38         Agua Pokemon11 = new Agua ("Gyarados", "Agua", 90);
39         this.listaPAE.añadirPokemon(Pokemon11);
40         Agua Pokemon12 = new Agua ("Nereidra", "Agua", 90);
41         this.listaPAE.añadirPokemon(Pokemon12);
42     }
43
44     public void imprimir() {
45         this.listaPAE.imprimir();
46     }
47
48     public void eliminarPokemon(Pokemon pPK) {
49         this.listaPAE.eliminarPokemon(pPK);
50     }
51
52     public Pokemon sacarPokemon(int pPosPK) throws Exception {
53         return this.listaPAE.sacarPokemon(pPosPK-1);
54     }
55 }
```

```
1 package org.pmoo.packproyecto;
2
3 public class Jugador {
4     private String nombre;
5     private int numCuras=3;
6     private ListaPokemon listaPokemon;
7
8     public Jugador(String pNombre) {
9         this.nombre=pNombre;
10        this.listaPokemon=new ListaPokemon();
11    }
12
13     public Pokemon sacarPokemon(int pPosPK) throws Exception{
14         return this.listaPokemon.sacarPokemon(pPosPK-1);
15     }
16
17     public boolean listaVacia() {
18         return this.listaPokemon.listaVacia();
19     }
20
21     public void imprimirLista() {
22         this.listaPokemon.imprimir();
23     }
24
25     public String getNombre() {
26         return this.nombre;
27     }
28
29     public boolean usarCura() {
30         boolean agotadas=false;
31         if(this.numCuras<=0) {
32             agotadas=true;
33             System.out.println("No te quedan curas.");
34         }
35         else {
36             this.numCuras--;
37         }
38         return agotadas;
39     }
40
41     public void imprimirNumCuras() {
42         System.out.println("\n6. USAR CURA (Te quedan "+this.numCuras+" curas).");
43     }
44
45     public void eliminarPokemon(Pokemon pPK) {
46         this.listaPokemon.eliminarPokemon(pPK);
47     }
48
49     public void añadirPokemon(Pokemon pPK) {
50         this.listaPokemon.añadirPokemon(pPK);
51     }
52
53     public void setNombre(String pNombre) {
54         this.nombre=pNombre;
55     }
56 }
```

```
3④ import static org.junit.Assert.*;
8
9 public class JugadorTest {
10
11     private Jugador j1;
12     private Agua pk1;
13     private Fuego pk2;
14     private Planta pk3;
15
16    @Before
17    public void setUp() throws Exception {
18        j1 = new Jugador("Jugador1");
19        pk1 = new Agua("PK1", "Agua", 10);
20        pk2 = new Fuego("PK2", "fuego", 3);
21        pk3 = new Planta("PK3", "Planta", 11);
22        j1.añadirPokemon(pk1);
23        j1.añadirPokemon(pk2);
24        j1.añadirPokemon(pk3);}
25    @After
26    public void tearDown() throws Exception {}
27
28    @Test
29    public void testSacarPokemon() {
30        Pokemon p;
31        try {
32            p = j1.sacarPokemon(5);
33            assertEquals(p,pk2);
34        } catch (Exception e) {}
35        assertThrows(Exception.class, ()->{j1.sacarPokemon(-1);});
36
37    @Test
38    public void testListaVacia() {
39        assertFalse(j1.listaVacia());
40
41    @Test
42    public void testGetNombre() {
43        assertEquals("Jugador1", j1.getNombre());
44
45    @Test
46    public void testEliminarPokemon() {
47        j1.eliminarPokemon(pk2);
48        Pokemon p;
49        try {
50            p = j1.sacarPokemon(5);
51            assertNotEquals(p,pk2);
52        } catch (Exception e) {}
53
54    @Test
55    public void testAñadirPokemon() {
56        Agua pk4 = new Agua("PK4", "Agua", 8);
57        j1.añadirPokemon(pk4);
58        Pokemon p;
59        try {
60            p = j1.sacarPokemon(7);
61            assertEquals(p,pk4);
62        }
63        catch (Exception e) {}
64
65    @Test
66    public void testSetNombre() {
67        assertEquals("Jugador1", j1.getNombre());
68        j1.setNombre("J1");
69        assertEquals("J1", j1.getNombre());
70    }
71 }
```

```

1 package org.pmoa.packproyecto;
2
3 public class ListaJugador {
4     private Jugador jugador1;
5     private Jugador jugador2;
6     private static ListaJugador miLJ=new ListaJugador();
7
8     private ListaJugador() {
9         this.jugador1=new Jugador(null);
10        this.jugador2=new Jugador(null);
11    }
12
13     public static ListaJugador getListaJugador() {
14         return miLJ;
15     }
16
17     public void imprimir(int pNumJugador) {
18         if(pNumJugador==1) {
19             this.jugador1.imprimirLista();
20         }
21         else if(pNumJugador==2) {
22             this.jugador2.imprimirLista();
23         }
24     }
25
26     public boolean listaVacia(int pPosJugador) {
27         boolean vacia=false;
28         if(pPosJugador==1&&this.jugador1.listaVacia()) {
29             vacia=true;
30         }
31         else if(pPosJugador==2&&this.jugador2.listaVacia()) {
32             vacia=true;
33         }
34         return vacia;
35     }
36
37     public String getNombre(int pNumJugador) {
38         String nombre=null;
39         if(pNumJugador==1) {
40             nombre=this.jugador1.getNombre();
41         }
42         else if(pNumJugador==2) {
43             nombre=this.jugador2.getNombre();
44         }
45         return nombre;
46     }
47
48     public Pokemon sacarPokemon(int pNumJugador, int pPosPK) throws Exception{
49         Pokemon pokemon=null;
50         if(pNumJugador==1) {
51             pokemon=this.jugador1.sacarPokemon(pPosPK);
52         }
53         else if(pNumJugador==2) {
54             pokemon=this.jugador2.sacarPokemon(pPosPK);
55         }
56         return pokemon;
57     }
58
59     public void añadirPokemon(int pNumJugador, Pokemon pPK) {
60         if(pNumJugador==1) {
61             this.jugador1.añadirPokemon(pPK);
62         }
63         else if(pNumJugador==2) {
64             this.jugador2.añadirPokemon(pPK);
65         }

```

```
66
67
68    public void eliminarPokemon(int pNumJugador, Pokemon pPK) {
69        if(pNumJugador==1) {
70            this.jugador1.eliminarPokemon(pPK);
71        }
72        else if(pNumJugador==2) {
73            this.jugador2.eliminarPokemon(pPK);
74        }
75    }
76
77    public void setNombres(String pNomJugador1, String pNomJugador2) {
78        this.jugador1.setNombre(pNomJugador1);
79        this.jugador2.setNombre(pNomJugador2);
80    }
81
82    public boolean usarCura(int pNumJugador) {
83        boolean agotada=false;
84        if(pNumJugador==1) {
85            agotada=this.jugador1.usarCura();
86        }
87        else if(pNumJugador==2) {
88            agotada=this.jugador2.usarCura();
89        }
90        return agotada;
91    }
92
93    public void imprimirNumCuras(int pNumJugador) {
94        if(pNumJugador==1) {
95            this.jugador1.imprimirNumCuras();
96        }
97        else if(pNumJugador==2) {
98            this.jugador2.imprimirNumCuras();
99        }
100    }
101 }
```

```

9  public class ListaJugadorTest {
10     Agua pk1,pk4;
11     Fuego pk2,pk5;
12     Planta pk3,pk6;
13     @Before
14     public void setUp() throws Exception {
15         ListaJugador.getListaJugador().setNombres("Jugador1","Jugador2");
16         pk1=new Agua("PK1", "Agua", 10);
17         pk2=new Fuego("PK2", "Fuego", 3);
18         pk3=new Planta("PK3", "Planta", 11);
19         ListaJugador.getListaJugador().añadirPokemon(1, pk1);
20         ListaJugador.getListaJugador().añadirPokemon(1, pk2);
21         ListaJugador.getListaJugador().añadirPokemon(1, pk3);
22         pk4=new Agua("PK4", "Agua", 10);
23         pk5=new Fuego("PK5", "Fuego", 3);
24         pk6=new Planta("PK6", "Planta", 11);
25         ListaJugador.getListaJugador().añadirPokemon(2, pk4);
26         ListaJugador.getListaJugador().añadirPokemon(2, pk5);
27         ListaJugador.getListaJugador().añadirPokemon(2, pk6);
28     }
29     @After
30     public void tearDown() throws Exception {}
31     @Test
32     public void testListaVacia() {
33         assertFalse(ListaJugador.getListaJugador().listaVacia(1));
34         assertFalse(ListaJugador.getListaJugador().listaVacia(2));}
35
36     @Test
37     public void testAñadirPokemon() {
38         try {
39             assertEquals(ListaJugador.getListaJugador().sacarPokemon(1,1),pk1);
40         } catch (Exception e) {}}
41
42     @Test
43     public void testGetNombre() {
44         assertEquals("Jugador1", ListaJugador.getListaJugador().getNombre(1));
45         assertEquals("Jugador2", ListaJugador.getListaJugador().getNombre(2));}
46
47     @Test
48     public void testSacarPokemon() {
49         try {
50             assertNotNull(ListaJugador.getListaJugador().sacarPokemon(1, 1));
51         } catch (Exception e) {}
52         try {
53             assertNotNull(ListaJugador.getListaJugador().sacarPokemon(2, 1));
54         } catch (Exception e) {}}
55
56     @Test
57     public void testEliminarPokemon() {
58         ListaJugador.getListaJugador().eliminarPokemon(1, pk2);
59         Pokemon p;
60         try {
61             p = ListaJugador.getListaJugador().sacarPokemon(1, 2);
62             assertNotEquals(p,pk2);
63         } catch (Exception e) {}}
64
65     @Test
66     public void testSetNombres() {
67         assertEquals("Jugador1", ListaJugador.getListaJugador().getNombre(1));
68         assertEquals("Jugador2", ListaJugador.getListaJugador().getNombre(2));
69         assertNotEquals("Jug1", ListaJugador.getListaJugador().getNombre(1));
70         assertNotEquals("Jug2", ListaJugador.getListaJugador().getNombre(2));
71     }
72 }
```

```
1 package org.pmoo.packproyecto;
2
3 public class Ataque {
4     private String nombre;
5     private String tipo;
6     private int daño;
7     private int usos;
8
9     public Ataque(String pNombre, String pTipo, int pDaño, int pUsos) {
10         this.nombre=pNombre;
11         this.tipo=pTipo;
12         this.daño=pDaño;
13         this.usos=pUsos;
14     }
15
16     public void imprimir(int pPosAtaqueEnLista) {
17         System.out.println(pPosAtaqueEnLista + ". NOMBRE : " + this.nombre + ", TIPO : " + this.tipo + ", DAÑO : " + this.daño + ", USOS : " + this.usos);
18     }
19
20     public int getDaño() {
21         return this.daño;
22     }
23
24     public void disminuirUso() {
25         this.usos--;
26     }
27
28     public boolean ataqueGastado() {
29         boolean gastado=false;
30         if(this.usos<=0) {
31             gastado=true;
32         }
33         return gastado;
34     }
35 }
```

```
1 package org.pmoa.packproyecto;
2
3④ import static org.junit.Assert.*;
4
5
6 public class AtaqueTest {
7
8     Ataque a1,a2,a3;
9
10    @Before
11    public void setUp() throws Exception
12    {
13        a1= new Ataque("Llama", "Fuego", 50, 4);
14        a2= new Ataque("Follaje", "Planta", 50, 1);
15        a3= new Ataque("Chorro", "Agua", 50, 0);
16    }
17
18    @After
19    public void tearDown() throws Exception {
20        a1 = null;
21        a2 = null;
22        a3 = null;
23    }
24
25    @Test
26    public void testAtaque() {
27        assertNotNull(a1);
28    }
29
30    @Test
31    public void testGetDaño() {
32        a1= new Ataque("Llama", "Fuego", 50, 4);
33        assertEquals(50,a1.getDaño());
34    }
35
36    @Test
37    public void testDisminuirUso() {
38        a2.disminuirUso();
39        assertTrue(a2.ataqueGastado());
40    }
41
42    @Test
43    public void testAtaqueGastado() {
44        assertFalse(a1.ataqueGastado());
45        assertTrue(a3.ataqueGastado());
46    }
47
48    }
49
50
51 }
```

```
1 package org.pmoa.packproyecto;
2
3+ import java.util.ArrayList;
4
5
6 public class ListaAtaque {
7     private ArrayList<Ataque> listaAtaque;
8
9+     public ListaAtaque() {
10         this.listaAtaque=new ArrayList<Ataque>();
11     }
12
13+     private Iterator<Ataque> getIterador(){
14         return this.listaAtaque.iterator();
15     }
16
17+     public void imprimir() {
18         Iterator<Ataque> itr=this.getIterador();
19         Ataque ataque;
20         int posEnLista=1;
21         System.out.println("\nLos ataques de tu Pokemon son :");
22         while(itr.hasNext()) {
23             ataque=itr.next();
24             ataque.imprimir(posEnLista);
25             posEnLista++;
26         }
27     }
28
29+     public int usarAtaque(int pPosAtaqueEnLista) {
30         Ataque ataque=this.listaAtaque.get(pPosAtaqueEnLista);
31         int daño=-100000;
32         if(!ataque.ataqueGastado()&&ataque!=null) {
33             daño=ataque.getDaño();
34             ataque.disminuirUso();
35         }
36         else{
37             System.out.println("\nSe han agotado los usos de este ataque.");
38         }
39         return daño;
40     }
41
42+     public void añadirAtaque(Ataque pAtaque) {
43         this.listaAtaque.add(pAtaque);
44     }
45 }
```

```

1 package org.pmoo.packproyecto;
2
3+import static org.junit.Assert.*;□
4
5 public class ListaAtaqueTest {
6
7     private ListaAtaque l1;
8
9     @Before
10    public void setUp() {
11        l1 = new ListaAtaque();
12    }
13
14    @After
15    public void tearDown() {
16        l1 = null;
17    }
18    @Test
19    public void testListaAtaque() {
20        assertNotNull(l1);
21    }
22
23    @Test
24    public void testUsarAtaque() {
25        Ataque a = new Ataque("PRUEBA", "TIP01", 100, 2);
26        l1.añadirAtaque(a);
27        assertEquals(100, l1.usarAtaque(0));
28        assertEquals(100, l1.usarAtaque(0));
29        assertEquals(-100000, l1.usarAtaque(0));
30    }
31
32    public void testAñadirAtaque(Ataque pAtaque) {
33        Ataque a1 = new Ataque("Pepe", "Agua", 50, 1);
34        l1.añadirAtaque(a1);
35        assertEquals(50, l1.usarAtaque(0));
36    }
37
38}
39
40}
41

```

```

1 package org.pmoa.packprojeto;
2 import java.util.Scanner;
3
4 public class Teclado {
5     private static Teclado miTEC=new Teclado();
6     private Scanner sc;
7     private Teclado() {
8         sc=new Scanner(System.in);}
9     public static Teclado getTeclado() {
10        return miTEC;
11    }
12    public String leerString() {
13        return sc.next();
14    }
15    public int leerNumero() {
16        int num=0;
17        boolean seguir=true;
18        do {
19            try {
20                seguir=false;
21                num=sc.nextInt();
22            }
23            catch(Exception e) {
24                sc.next();
25                seguir=true;
26            }}}}while(seguir);
27        return num;}
28
29    public int leerNumero(int pMax) {
30        int num=0;
31        boolean seguir=true;
32        do {
33            try {
34                seguir=false;
35                num=sc.nextInt();
36                while(num>pMax||num<0) {
37                    System.out.println("ERROR: Su respuesta no es el esperada.");
38                    num=sc.nextInt();
39                }
40            catch(Exception e) {
41                sc.next();
42                seguir=true;
43            }}}} while(seguir);
44        return num;
45    }
46    public boolean leerSiNo() {
47        boolean sino=false;
48        boolean hecho=false;
49        String respuesta;
50        while(!hecho) {
51            respuesta=sc.next().toLowerCase();
52            if(respuesta.equals("sí") || respuesta.equals("si")) {
53                sino=true;
54                hecho=true;}
55            else if(respuesta.equals("no")){
56                sino=false;
57                hecho=true;}
58            else {
59                System.out.println("ERROR: Su respuesta no es la esperada");
60            }
61        }
62        return sino;
63    }

```

```

1 package org.pmoo.packproyecto;
2
3 import java.util.Random;
4
5 public class Tablero {
6     private Pokemon pokemon1;
7     private Pokemon pokemon2;
8     private static Tablero miTAB=new Tablero();
9
10    private Tablero() {}
11
12    public static Tablero getTablero() {
13        return miTAB;
14    }
15
16    public void empezarPartida() {
17        System.out.println("¿Quieres empezar una partida? (Escribe 'Si' o 'No')");
18        boolean respuesta=Teclado.getTeclado().leerSiNo();
19        if(respuesta) {
20            partida();
21        }
22        else {
23            System.out.println(";;Esperamos que vuelvas pronto!!!");
24        }
25    }
26
27    private void partida() {
28        System.out.println("COMIENZA LA PARTIDA");
29        crearJugadores();
30        try {
31            this.pokemon1=ListaJugador.getListajugador().sacarPokemon(1,this.generarNumRandom(3));
32            this.pokemon2=ListaJugador.getListajugador().sacarPokemon(2,this.generarNumRandom(3));
33        }
34        catch(Exception e) {}
35        while(!ListaJugador.getListajugador().listavacia(1)&&!ListaJugador.getListajugador().listavacia(2)) {
36            this.atacar();
37        }
38        if(ListaJugador.getListajugador().listavacia(1)) {
39            System.out.println("EL GANADOR ES "+ListaJugador.getListajugador().getNombre(2));
40        }
41        else if(ListaJugador.getListajugador().listavacia(2)) {
42            System.out.println("EL GANADOR ES "+ListaJugador.getListajugador().getNombre(1));
43        }
44    }
45
46    private void crearJugadores() {
47        System.out.println("\nEscribe el nombre del Jugador1");
48        String nombre1=Teclado.getTeclado().leerString();
49        System.out.println("Escribe el nombre del Jugador2");
50        String nombre2=Teclado.getTeclado().leerString();
51        ListaJugador.getListajugador().setNombres(nombre1, nombre2);
52        this.elegirListaPokemon();
53    }
54
55    private void elegirListaPokemon() {
56        System.out.println("ELIGE LOS TRES POKEMON QUE QUIERAS UTILIZAR.");
57        int terminar=0;
58        while(terminar<3) {
59            ListaPokemonAElegir.getListapokemonAElegir().imprimir();
60            this.elegirPokemon(1);
61            ListaPokemonAElegir.getListapokemonAElegir().imprimir();
62            this.elegirPokemon(2);
63            terminar++;
64        }
65    }

```

```

67
68     private void elegirPokemon(int pPosJug) {
69         System.out.println("\nTE TOCA ELEGIR "+ListaJugador.getListajugador().getNombre(pPosJug));
70         int num=Teclado.getTeclado().leerNumero(12);
71         try {
72             Pokemon pk=ListaPokemonAElegir.getListapokemonAElegir().sacarPokemon(num);
73             ListaPokemonAElegir.getListapokemonAElegir().eliminarPokemon(pk);
74             ListaJugador.getListajugador().añadirPokemon(pPosJug, pk);
75         }
76         catch(Exception e) {
77             System.out.println("ERROR: Su respuesta no es la esperada.");
78             this.elegirPokemon(pPosJug);
79         }
80     }
81
81     private int generarNumRandom(int pMax) {
82         Random r=new Random();
83         return r.nextInt(pMax)+1;
84     }
85
85     private void atacar() {
86         try {
87             System.out.println("\n¿Qué ataque quieras usar "+ListaJugador.getListajugador().getNombre(1)+"?");
88             System.out.println("\n                                         CONTRINCANTE");
89             this.pokemon2.imprimirSoloInfoPK();
90             this.pokemon1.imprimir();
91             System.out.println("\n5.   CAMBIAR DE POKEMON");
92             ListaJugador.getListajugador().imprimirNumCuras(1);
93             this.pokemon1.atacar(pokemon2);
94             if(this.pokemon2.muerto()) {
95                 ListaJugador.getListajugador().eliminarPokemon(2, this.pokemon2);
96                 this.pokemon2=null;
97                 while(this.pokemon2==null&&!ListaJugador.getListajugador().listavacia(1)&&!ListaJugador.getListajugador().listavacia(2)) {
98                     this.muerto(2);
99                 }
100            }
101        }
102        catch(CambiarDePokemonException cdpe){
103            this.cambiarPokemon(1);
104        }
105        catch(UsarCuraException uce) {
106            if(!ListaJugador.getListajugador().usarCura(1)) {
107                this.pokemon1.cambiarVida(-15);
108            }
109        }
110    try {
111        if(!ListaJugador.getListajugador().listavacia(1)&&!ListaJugador.getListajugador().listavacia(2)) {
112            System.out.println("\n¿Qué ataque quieras usar "+ListaJugador.getListajugador().getNombre(2)+"?");
113            System.out.println("\n                                         CONTRINCANTE");
114            this.pokemon1.imprimirSoloInfoPK();
115            this.pokemon2.imprimir();
116            System.out.println("\n5.   CAMBIAR DE POKEMON");
117            ListaJugador.getListajugador().imprimirNumCuras(2);
118            this.pokemon2.atacar(pokemon1);
119            if(this.pokemon1.muerto()&&!ListaJugador.getListajugador().listavacia(1)&&!ListaJugador.getListajugador().listavacia(2)) {
120                ListaJugador.getListajugador().eliminarPokemon(1, this.pokemon1);
121                this.pokemon1=null;
122                while(this.pokemon1==null&&!ListaJugador.getListajugador().listavacia(1)&&!ListaJugador.getListajugador().listavacia(2)) {
123                    this.muerto(1);
124                }
125            }
126        }
127    }
128    catch(CambiarDePokemonException cdpe) {
129        this.cambiarPokemon(2);
130    }
131    catch(UsarCuraException uce) {
132        if(!ListaJugador.getListajugador().usarCura(2)) {
133            this.pokemon2.cambiarVida(-15);
134        }
135    }
136 }
137
138

```

```

139@ private void muerto(int pPosJugador) {
140     System.out.println("Ha muerto tu pokemon " + ListaJugador.getListaJugador().getNombre(pPosJugador) + " ¿Que pokemon quieres sacar ahora?");
141     ListaJugador.getListaJugador().imprimir(pPosJugador);
142     boolean parar=false;
143     while(!parar) {
144         int num=Teclado.getTeclado().leerNumero();
145         if(pPosJugador==1) {
146             try {
147                 this.pokemon1=ListaJugador.getListaJugador().sacarPokemon(pPosJugador,num);
148                 parar=true;
149             }
150             catch(Exception e) {
151                 System.out.println("ERROR: Su repuesta no es la esperada.");
152             }
153         }
154         else if(pPosJugador==2) {
155             try {
156                 this.pokemon2=ListaJugador.getListaJugador().sacarPokemon(pPosJugador,num);
157                 parar=true;
158             }
159             catch(Exception e) {
160                 System.out.println("ERROR: Su respuesta no es la esperada.");
161             }
162         }
163     }
164 }
165
166@ private void cambiarPokemon(int pPosJugador) {
167     System.out.println("¿Qué pokemon de tu lista quieres sacar?");
168     ListaJugador.getListaJugador().imprimir(pPosJugador);
169     boolean parar=false;
170     while(!parar){
171         try {
172             int num=Teclado.getTeclado().leerNumero(3);
173             if(pPosJugador==1){
174                 this.pokemon1=ListaJugador.getListaJugador().sacarPokemon(1,num);
175                 parar=true;
176             }
177             else if(pPosJugador==2) {
178                 this.pokemon2=ListaJugador.getListaJugador().sacarPokemon(2,num);
179                 parar=true;
180             }
181         }
182         catch(Exception e) {
183             System.out.println("ERROR: Su respuesta no es la esperada.");
184         }
185     }
186 }
187
188@ public static void main(String[]args) {
189     miTAB.empezarPartida();
190 }
191 }

```

```
1 package org.pmoo.packproyecto;
2
3 public class CambiarDePokemonException extends Exception{
4
5     public CambiarDePokemonException() {
6         super();
7     }
8 }
```

---

```
1 package org.pmoo.packproyecto;
2
3 public class UsarCuraException extends Exception{
4     public UsarCuraException() {
5         super();
6     }
7 }
```