with Ada.Unchecked\_Deallocation;

with Ada.Text\_IO;               use Ada.Text\_IO;

package body Piles is

    procedure Free is

        new Ada.Unchecked\_Deallocation (T\_Cellule, T\_Pile);

                        --TESTS--

    procedure Initialiser (*Pile* : out T\_Pile) is

    begin

        Pile := Null;

    end Initialiser;

    function Est\_Vide (*Pile* : in T\_Pile) return Boolean is

    begin

        return Pile = Null;

    end Est\_Vide;

    function Existe\_Pile(*Element*: in T\_Element; *Pile*: in T\_Pile) return Boolean is

        Parcours:T\_Pile:=Pile;

        begin

            while Parcours/=Null loop

                if Sommet(Parcours)=Element then

                    return True;

                end if;

                Parcours:=Parcours.all.Suivant;

            end loop;

            return False;

        end Existe\_Pile;

                --FONCTIONS ELEMENTAIRES--

    function Sommet (*Pile* : in T\_Pile) return T\_Element is

    begin

        if Pile/=Null then

            return Pile.all.Element;

        else

            raise Pile\_Vide;

        end if;

    end Sommet;

    function Next\_Pile(*Pile*: in T\_Pile) return T\_Pile is

        begin

            if Pile=Null then

                return Null;

            else

                return Pile.all.Suivant;

            end if;

        end Next\_Pile;

    function Size\_Pile(*Pile*: in T\_Pile) return Integer is

        Parcours:T\_Pile;

        i:Integer:=0;

        begin

            if Est\_Vide(Pile) then

                return 0;

            else

                Parcours:=Pile;

                while not Est\_Vide(Parcours) loop

                    i:=i+1;

                    Parcours:=Parcours.all.Suivant;

                end loop;

                return i;

            end if;

        end Size\_Pile;

                  --AJOUT/SUPPRESSION--

    procedure Empiler (*Pile* : in out T\_Pile; *Element* : in T\_Element) is

        Nouvelle\_Cellule: T\_Pile;

    begin

        Nouvelle\_Cellule := new T\_Cellule;

        Nouvelle\_Cellule.all.Element := Element; --Nouvelle cellule contenant l'élément

        Nouvelle\_Cellule.all.Suivant := Pile; --Rattacher le "bout" de cette cellule avec le reste de la pile

        Pile := Nouvelle\_Cellule; --La pile contient maintenant l'élément empilé ainsi que le reste de la pile

    end Empiler;

    procedure Depiler (*Pile* : in out T\_Pile) is

        A\_Detruire : T\_Pile;

    begin

        A\_Detruire := Pile;

        Pile := Pile.all.Suivant; --Le sommet de la pile est maintenant le second élément

        Free (A\_Detruire); --Detruction de l'ancien sommet

    end Depiler;

    procedure Supprimer\_Element(*Element*: in T\_Element; *Pile*: in out T\_Pile) is

        Parcours,A\_Supprimer: T\_Pile;

        begin

            if Pile=Null then

                Null;

            elsif not Existe\_Pile(Element, Pile) then

                Null;

            else

                if Sommet(Pile)=Element then

                    Depiler(Pile);

                else

                    Parcours:= Pile;

                    while Parcours.all.Suivant/=Null and then Sommet(Parcours.all.Suivant)/=Element loop

                        Parcours:=Parcours.all.Suivant;

                    end loop;

                    A\_Supprimer:=Parcours.all.Suivant;

                    Parcours.all.Suivant:=Parcours.all.Suivant.all.Suivant;

                    Free(A\_Supprimer);

                end if;

            end if;

        end Supprimer\_Element;

    procedure Detruire (*P*: in out T\_Pile) is

    begin

        if P /= Null then

            Detruire (P.all.Suivant);

            Free (P); --Detruction de la pile à partir du premier élément empilé jusqu'au sommet

        else

            Null;

        end if;

    end Detruire;

    procedure Affecter\_Pile(*P*:in out T\_Pile;*P2*: in T\_Pile) is

        begin

            P:=P2;

        end Affecter\_Pile;

                    --AFFICHAGE--

    procedure Afficher\_Pile (*Pile* : in T\_Pile) is

        procedure Afficher\_Elements (*Pile* : in T\_Pile) is

        begin

            if Pile = Null then

                Null;

            elsif Pile.all.Suivant = Null then

                Put (" ");

                Afficher\_Element (Pile.all.Element);

            else

                Afficher\_Elements (Pile.all.Suivant);

                Put (", ");

                Afficher\_Element (Pile.all.Element);

            end if;

        end Afficher\_Elements;

    begin

        Afficher\_Elements (Pile);

        Put (" ."); --La pile sera affichée comme W, X, Y, Z .

    end Afficher\_Pile;

end Piles;