|  |  |  |
| --- | --- | --- |
|  |  | **جامعة دمشق**  **كلية الهندسة المعلوماتية**  **سنة 3** |
| **تقرير وظيفة لغات البرمجة** | | |
|  |  | **إعداد الطلاب:** |
| **وسام الرجولة** | **عمار الجزائرلي** | **عبد الهادي بيطار** |
|  |  |  |

**التمرين الأول:**

1 . اسم الملف: Variables موجود ضمن package name :variavles التعديلات تمت على ال project

2. بعد الدخول للشاشة السوداء و الوصول إلى المجلد الحاوي على ال package ننفذ التعليمات

>javac variavles \Variables.java

التعليمة تشغل المترجم الذي يعطي الملف Variables.class

>java variavles\Variables

التعليمة تشغل المترجم الفوري ويظهر الخرج على الشاشة السوداء

3 . لأنه متحول Static ويمكن الوصول اليه عن طريق اسم ال class دون الحاجة لتعريف object

تم الاسناد بوساطة *setCLASSVARIABLE*(1);

قبل الاسناد قيمة الخرج 0 , لأن classVariables من النوع int تأخذ قيمة افتراضية 0

بعد الاسناد قيمة الخرج 1 وهي القيمة التي تم اسنادها

4 . لأنه متحول يتم انشاءة عند بناء object ولايمكن استدعاء الا عن طريق اسم الا object

تم الاسناد بواسطة

Variables testVariable = **new** Variables();

testVariable.setInstanceVariable(1);

وعند استدعاءه اول مرة قبل الاسناد يرد قيمة 0 لأنها قيمة ابتدائية لمتحولات int ((primitive type وبعد الاسناد القيمة التي تم اسنادها

5 . لأنه متحول ضمن method الاسناد مباشر ولا يمكن طبع قيمته قبل اسناده لأنه لا يأخذ قيمة ابتدائيةوطريقة الطباعة مباشرة عن طريق اسم المتحول

6 . يمكن امام المتحول instancevariable ولكنه سيتحول الى classvariable

ولكن يجب تحول التاوبع Setter and getter الى توابع static ويتم استدعاءه عن طريق اسم ال class وليس object ولا يمكن وضع Static امام localvariable لأنه يا يمكن تعريف متحول ٍستاتيكي ضمن method

7 . لا توجد مشكلة بالنسبة للمتحولات class and instance variable

لأنه عند التنفيذ يتم تعريف class member وال Constructer

ومن ثم يبدأ بتنفيذ ال main method ولكن بالنسبة ل localVariable اولا لايمكن اسناد قيمة له بدون تعريفه ثانيا اذا عرفناه واسندا له قيمة لا يكمن اعادة تعريفه مرة ثانية ولا يمكن ان تتم عملية الطباعة قبل عملية التعريف والاسناد لأن البرنامج يعمل تعليمة تعليمة داخل ال method

8 . يمكن اضافة final لكل من المتحولات الثلاث ولكن مع شروط

للمتحول classVariable : يمكن وضع كلمة final على الشكل التالي

**final** **private** **static** **int** *CLASSVARIABLE* ;

ولكن يجب اعطاءه قيمة ابتدايئة مباشرة

للمتحول instanceVariable : يمكن وضع كلمة final على الشكل التالي

**final** **private** **int** instanceVariable;

ولكن يجب اعطاءه قيمة ابتدايئة مباشرة أو ضمن ال constructer

للمتحول localVariable : يمكن وضع كلمة final على الشكل التالي

final int localVariavle ;

ويمكن اعطاءه قيمة ابتدائية مباشرة او في اي مكان اخر وسوف تصبح قيمة دائمة له لايمكن تغيرها

**التمرين الثاني:**

1. لأننا نقوم بتحديد نمط المتحول أي اننا نقوم بإخبار ال compiler ما هو نمط المعطيات المراد للمتحول   
   مثال :

Int x = 3 ;

اما ال loosely typed languages يكون الحجز دون تحديد نمط معين

مثال in javascript:

var marks = 20 ;

var str =”1dfgdfgdfg” ;

var raining = true ;

|  |  |
| --- | --- |
| الأنماط البسطٌية | والأنماط المرجعيةٌ |
| تحمل قيم | تحمل مرجع |
| لا يمكن أن تكون null | يمكن أن تكون null |

boolean **,**char ,byte ,short ,int ,long ,float ,double ,void

1. نعم للاستفادة من مزايا الأنماط المرجعية (وجود توابع جاهزة لمعظم العمليات التي نحتاجها )
2. نعم

Integer.parseInt("123")=123;

Integer.parseInt("123",8)=83

Integer.parseInt("123",16)=291

بالمثل

Short.parseShort()

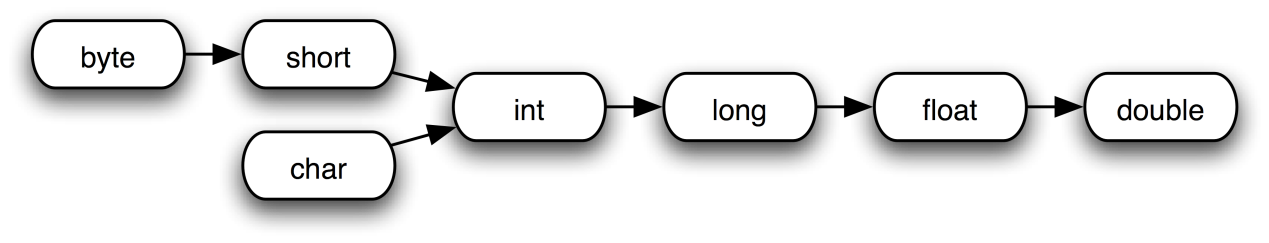
Byte.parseByte()

Double.parseDouble("5")=5.0

Double.parseDouble("5.5")=5.5

بالمثل

Float.parseFloat()



تقوم الجافا تلقائياً بالتحويل بين الأنماط كما هو موضح عند الاسناد

int num='s';

float fnum=1236547890;

Integer.MAX\_VALUE;= 2147483647

(float) Integer.MAX\_VALUE;= 2.14748365E9

Long.MAX\_VALUE= 9223372036854775807

(float) Long.MAX\_VALUE;= 9.223372E18

(double) Long.MAX\_VALUE;= 9.223372036854776E18

1. هناك نوعين من ال casting (up and down casting)

examples for downcasting : int x = 19.3 ; x ==19

ولكنه نوع غير آمن

يمكن استخدام ال downcasting على ال References ك قصر مؤشر من نمط ال superclass على subclass

Examples for upcasting : float y = x ;y == 19.0

كما يمكن استخدام ال upcasting على references ك استخدام مؤشر من نمط superclass ك يشير على غرض من نمط subclass

**التمرين الثالث:**

1. نستورد الصف القارئ java.util.Scanner إلى البرنامج من خلال استخدام التعليمة:



1. يغلف الدخل القياسي System.in بغرض من الصف القارئ من خلال استخدام السطر التالي:



1. قراءة قيم لمتحولات بأنماط مختلفة باستخدام القارئ:



1. التأكد من الدخل من أنه يوافق نمطا معينا:



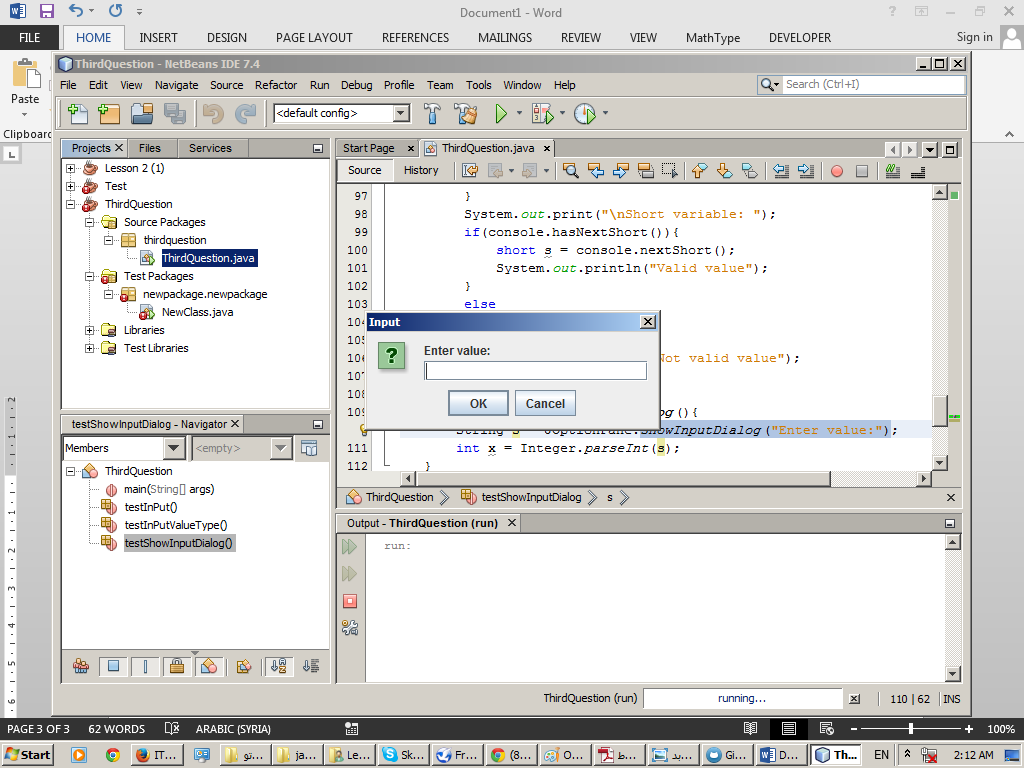
1. نستورد الصفjavax.swing.JOptionPane من خلال استخدام التعليمة:



1. القيمة المعادة من الإجرائية showInputDialog هي من نوع String مثال:



الخرج



يتم تحويل القيمة إلى نمط ما مثل int وفق المثال:



**التمرين الرابع:**

1. لا ,التعليل لأن ال package java.long مضمنة بشكل default مع ال project
2. لا ,التعليل : كل التوابع و الثوابت التي نحتاجها هي public static

try{

System.out.println("enter cirule radius \n R = ");

Scanner in = new Scanner(System.in);

float r = in.nextFloat();

if(r<0) throw new Exception("R is less than zero");

System.out.println("\n2\*pi\*R = " + 2\*Math.PI\*r

+ "\npi\*r^2 = " + Math.PI\*Math.pow(r, 2));

}

catch(Exception e){

System.err.println(e.getMessage());

}

1. A:

Min + (int)(Math.random() \* ((Max - Min) + 1))

B:

try{

System.out.println("enter throwing times \n TT = ");

Scanner in = new Scanner(System.in);

int n = in.nextInt();

if(n<1) throw new Exception("throwing times is less than 1");

for (int i = 0; i < n; i++) {

System.out.println(1 + (int)(Math.random() \* 6));

}

}

catch(Exception e){

System.err.println(e.getMessage());

}

**التمرين الخامس:**

|  |  |
| --- | --- |
| 2 . public Product (int productID , String ProductName){  setProductID(productID);  setProductName(ProductName);  InstanceCount++;  }  3 .  public Product (int productID , String productName , float price ,int quanity){  this(productID,productName) ;  setPrice(price);  setQuanity(quanity);  } | 1 . public class Product implements Comparable<Product>{    private int productID ;  public int getProductID() {  return productID;  }  public void setProductID(int productID) {  this.productID = productID;  }    private String productName ;  public String getProductName() {  return productName;  }  public void setProductName(String productName) {  this.productName = productName;  }    private float price ;  public float getPrice() {  return price;  }  public void setPrice(float price) {  this.price = price;  }  private int quanity ;  public int getQuanity() {  return quanity;  }  public void setQuanity(int quanity) {  this.quanity = quanity;  } |

1. A: لا لاتمكننا لأنها تقارن بين المؤشرات وليس الأغراض

B: لا لأننا لم نقم بإعادة تعريف التابع بعد وهوة يأخذ التابع من الأب object وهوة يقارن بين المؤشرات أيضا

|  |  |
| --- | --- |
| if (!Objects.equals(this.productName, other.productName)) {  return false;  }  if (Float.floatToIntBits(this.price) != Float.floatToIntBits(other.price)) {  return false;  }  if (this.quanity != other.quanity) {  return false;  }  return true;  } | @Override  public boolean equals(Object obj) {  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  final Product other = (Product) obj;  if (this.productID != other.productID) {  return false;  } |

C: لا لايمكننا لأننا لم نحدد معيار المقارنة للترتيب من خلال اعادة تعريف التابع comperTo

D:

public static Comparator<Product> getPoductNameCorparator(){

return new Comparator<Product>() {

@Override

public int compare(Product o1, Product o2) {

return o1.getProductName().compareTo(o2.getProductName()) ;

}

} ;

}