**Topshiriq-1. Nazariy savollar uchun namunalar**

**1. Javada ma’lumotlar tiplari va o’zgaruvchilarni e’lon qilish qonuniyatlari. Misollar keltiring**

**Javob:**

**1 ta simvol uchun o’zgaruvchi char**

**Simvollar ketma ketligi uchun o’zgaruvchi String**

**Sonlar uchun o’zgaruvchi int**

**Butun sonlar uchun o’zgaruvchi Float(verguldan keyin 6ta son aniqlikda)**

**Butun sonlar uchun o’zgaruvchi Double(verguldan keyin 1ta son aniqlikda)**

**– boolean(1)**

**– byte(8)**

**– short(16)**

**– int(32)**

**– long(64)**

**– char(16)**

**– float(32)**

**– double(64)**

**Boolean bool=true/false;**

**Int son=12;**

**Float raqam;**

**Raqam=12;**

**2. Javada kiritish va chiqarish operatorlaridan foydalanish. Kiritsh va chiqarish funksiyalarining qo’shimcha imkoniyatlari. Misollar keltiring**

**Javob: Chiqarish operatorlari:**

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

**System.out.print();**

**System.out.printf();**

**Kiritish operatorlari:**

**System.out.ReaD();**

**Scanner o’zgaruvchi=new Scanner(System.in)**

**3. Javada shart va tanlash operatorlaridan foydalanish. Dasturda if, if...else, switch...case bloklaridan foydalanish. Misollar keltiring**

**Javob:**

**If operatori:**

|  |  |  |
| --- | --- | --- |
| **if (shart) {**  **buyruqlar**  **}** | **if (shart) {**  **buyruqlar**  **} else {**  **buyruqlar**  **}** | **if (shart) {**  **// buyruqlar**  **} else if (shart2) {**  **// buyruqlar**  **}**  **...**  **else {**  **// buyruqlar**  **}** |

**Switch operatori:**

**switch (ifoda) {**

**case qiymat\_1 :**

**buyruqlar;**

**break;**

**. . .**

**case qiymat\_n :**

**buyruqlar;**

**break;**

**default:**

**buyruqlar;**

**}**

**Тернар амалдан фойдаланиш (?:)**

**Quyidagi shartni:**

**if (a > b) {**

**max = a;**

**} else {**

**max = b;**

**}**

**Quyidagi ko’rinishda yozish mumkin:**

**max = (a > b) ? a : b;**

**4. Javada sikl operatorlaridan foydalanish. Cheksiz sikl va ichma-ich sikl yaratish. Misollar keltiring**

**Javob:**

**Sikl Operatorlar**

**for ( shart; qadam ) {**

**buyruqlar;**

**}**

**while (shart) {**

**buyruqlar;**

**}**

**do {**

**buyruqlar;**

**} while (shart);**

**Мисол-4. for оператори учун**

**public class MainClass {**

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

**for (int i = 0; i < 5; i++) {**

**System.out.println(i + " ");**

**}**

**}**

**}**

**for (int i = 1, j = 100; i < 100; i = i+1, j = j-1) {**

**System.out.println(i + j);**

**}**

**Мисол-5. while оператори учун**

**public class WhileLoop {**

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

**int count = 1;**

**System.out.println(“1 - 10");**

**while (count <= 10) {**

**System.out.println(count++);**

**}**

**}**

**}**

**Мисол-6. do…while оператори учун**

**public class DoWhileLoop {**

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

**int count = 1;**

**System.out.println(“1 - 10");**

**do {**

**System.out.println(count++);**

**} while (count <= 10);**

**}**

**}**

**Cheksiz siklga misollar:**

**While(true){**

**Buyruqlar;**

**}**

**For( ; ; ){**

**Buyruqlar;**

**}**

**5. Javada massivlar. Bir o’lchovli va ikki o’lchovli massivlar yaratish. Misollar keltiring**

**Javob:**

**Massiv indeksli o’zgaruvchidir.**

**Massiv sodda ta’rifi:**

**<type> <var\_name>[<konstanta\_ifoda>] = <init>;**

**Massiv indekslar qiymati har doim 0 dan boshlanadi.**

**Massivga hotira ajratish uchun maxsus new operatoridan foydalaniladi. Massiv inisalizatsiya qilinganda elementlar soni ko’rsatilishi shart emas.**

**Masalan:**

**double d[] = {1, 2, 3, 4, 5};**

**Masalan:**

**xTu = new int[]{ 17, 19, 23, 29, 31, 37 };**

**Massiv elementlar sonini hisoblash uchun length dan foydalanish kerak.Massivlarni e’lon qilishda[] qovuslar ishlatiladi, [] qovuslar o’zgaruvchi toifasidan keyin qo’yilishi yoki o’zgaruvchi nomidan**

**keyin qo’yilishi mumkin:**

**Masalan:**

**byte[] x, y, z[];**

**Quyidagiga ekvivalent:**

**byte x[], y[], z[][];**

**Massivli o’zgaruvchilarni e’lon qilish:**

**int[] ai; // int toifali massiv**

**short[][] as; // massiv ichida massiv (array of array)**

**Object[] ao, // Object toifali massiv**

**short s, // oddiy o’zgaruvchi**

**aas[][]; // massiv ichida massiv (array of array)**

**Massivli o’zgaruvchilarni e’lon qilish va massivga keltirish:**

**Exception ae[] = new Exception[3];**

**Object aao[][] = new Exception[2][3];**

**int[] factorial = { 1, 1, 2, 6, 24, 120, 720, 5040 };**

**char ac[] = { 'n', 'o', 't', ' ', 'a', ' ',**

**'S', 't', 'r', 'i', 'n', 'g' };**

**String[] aas = { "array", "of", "String” };**

**Massivlar yaratish:**

**int d[]={1,2,3,4,5,6};**

**int d[]=new int[6];**

**1 o’lchovli massiv yaratish:**

**int[] a; // int toifali massiv bo’lishini e’lon qilish**

**a = new int[100]; // int uchun 100 ta elementga Xotiradan joy ajratish**

**Yuqoridagini quyidagi ko’rinishda yozish mumkin:**

**int[] a = new int[100]; // E’lon qilish va joy ajratish**

**1 o’lchamli massivga misol:**

**public class MainClass {**

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

**int[] intArray = new int[10];**

**for (int i = 0; i < 10; i++) {**

**intArray[i] = 100;**

**}**

**for (int i = 0; i < 10; i++) {**

**System.out.println(intArray[i]);**

**}**

**}**

**}**

**Ko’p o’lcho’vli massiv:**

**int a[][] = new int[2][4]**

**a[0][0] = 9;**

**a[0][1] = 5;**

**a[0][2] = 3;**

**a[1][3] = 7;**

**a[1][4] = 2;**

**Ko’p o’lcho’vli massivga misol:**

**class MultiDimArray {**

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

**int[] a[], b[];**

**b = new int[][] {{5,8,1},{4,5,6},{4,6,8}};**

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

**System.out.println(b[i][i]);**

**}**

**}**

**}**

**6. Javada satrlar bilan ishlash. String va StringBuilder sinfidan foydalanish, qo’shimcha metodlari. Misollar keltiring.**

**7. Javada sinflar va obyektlar. Sinf atributlari va metodlarini yaratish. Sinf konstruktorlari. Misollar keltiring.**

**Javob:**

**Obyektlar:**

**Sinfning obyekt yaratish ketma ketligi:**

**Declaration –Obyektni e’lon qilish;**

**Instantiation – new kalit so’zidan foydalanish;**

**Initialization – Obyektga dastlabki qiymat berish(new kalit so’zidan keyin konstruktorni chaqirish).**

**Javda obyektlar xususiyatlar(states) va metodlar(behaviors) dan tashkil topadi.**

**Sinflar(Classs)**

**Sinflar-obyektlardan tashkil topgan.**

**Javada sinflar quyidagi o’zgaruvchilardan tashkil topishi mumkin:**

**Local variables – bu faqat metod yoki konstruktorlar ichida yaratiladi va e’lon qilinadi.Va shu blok ichida ishlidi.**

**Intance variables-Bu sinf ichida lekin metodlar tashqarida e’lon qilinadi va bu o’zgaruvchilarga barcha metodlar va konstruktorlar murojaat qila oladi. Bunda this kalit so’zidan foydalanadi. Class variables-Bu sinf ichida lekin metodlar tashqarida e’lon qilinadigan static o’zgaruvchidir.**

**Javada sinf yaratishning umumiy ko’rinishi.**

**package dir.subdir // Sinf joylashgan papka**

**import java.classes.ClassName // Mavjud sinfni import qilish**

**class class\_name {**

**variables**

**methods() {**

**}**

**}**

**Konstruktorlar:**

**1.Sinf chaqirilganda ya’ni joriy sinfning obyekti yaratilganda ishga tushadi.**

**2.Nomi sinf nomi bilan bir xil bo’ladi.**

**3.Sinf metodi hisoblanadi.**

**4.Sinfda konstruktorlar bir nechta bo’lishi mumkin ya’ni konstruktorni overload qilsa bo’ladi.**

**Мисол:**

**public class Phone{**

**public Phone(){**

**}**

**public Phone(String name){**

**// This constructor has one parameter, name.**

**}**

**}**

**8. Obyektga yo’naltirilgan dasturlashda encapsulation tamoyili. Dasturda public, private va protected kalit so’zlaridan foydalanish va ularning farqlari. Misollar keltiring**

**Javob:**

**Sinflar atributlarini yashirish, boshqa sinflar tomonidan atributlarga to’g’ridan-to’g’ri murojaatni cheklash uchun obyektga yo’naltirilgan dasturlashda – encapsulation jarayoni mavjud.**

**Java dasturlash tilida encapsulation uchun ishlatiladigan quyidagi kalit so’zlar mavjud:**

**– private**

**– protected**

**– public (ochiq atribut, bunda encapsulation amalga oshirilmaydi, lekin atributga murojaat ochiqligini bildiradi)**

**– default (yuqoridagi kalit so’zlar yozilmagan holati)**

**Sinfda getter va setter metodlarni yaratish orqali read-only yoki write-only ko’rinishidagi sinflarni yaratish mumkin.**

**Sinf atributiga o’zlashtirilayotgan qiymatni tekshirish va nazorat qilish mumkin, ya’ni setter metodlar ichiga maxsus logika yozish mumkin.**

**Private kalit so’zi yordamida sinf atributini yashirish mumkin va bu orqali boshqa sinf yoki sinf obyektlari mazkur atributga murojaat qila olmaydi.**

**Odatda bironta obyektga yo’naltirilgan dasturlash tilida frameworklar yaratish uchun encapsulation ga ko’proq ehtiyoj seziladi.**

**9. Obyektga yo’naltirilgan dasturlashda sinflar orasida munosabatlar o’rnatish. Munosabat turlari, sinflarda HAS-A, PART-OF bog’lanish turlari. Misollar keltiring**

**Javob:**

**Sinflar orasida munosabatlar (association**

**HAS-A musosabat (aggrigation)**

**PART-OF munosabat (composition)**

**Sinflar o’rtasida munosabatlarni o’rnatish uchun quyidagi tushunchalarda foydalaniladi**

**– Association**

**– Aggrigation**

**– Composition**

**Assotsiativ munosabat (Association) - bu obyektlar orqali o'rnatiladigan ikkita alohida sinf o'rtasidagi munosabatlar.**

**■ Obyektlarning bir-biriga bog’liqligiga qarab quyidagi ko’rinishlarda bo’lishi mumkin:**

**– One-to-one (birga-bir)**

**– One-to-many (birga-ko’p)**

**– Many-to-one (ko’pga-bir)**

**– Many-to-many (ko’pga-ko’p)**

**■ Obyektlarning bog’lanish xususiyatiga qarab esa assotsiativ munosabatlar quyidagi 2 turga bo’linadi:**

**– Aggrigativ munosabat (aggrigation)**

**– Kompozitiv munosabat (composition)**

|  |  |  |
| --- | --- | --- |
| **Farqli xususiyati** | **Aggregation** | **Composition** |
| **Dependency** | **Bunda munosabat o’rnatilgan**  **sinflar (parent-child munosabat)**  **o’zaro mustaqil ravishda ishlatilishi**  **mumkin. Misol uchun: Bank**  **(parent class) va Ishchi (child class)**  **sinflari o’rtasida munosabat bo’lsa,**  **Bank obyektini o’chirib tashlasa**  **ham Ishchi sinfining obyekti**  **mavjud bo’la oladi.** | **Bunda munosabat o’rnatilgan**  **sinflar (parent-child) o’zaro**  **mustaqil ravishda ishlatilishi**  **mumkin emas. Misol uchun Car**  **(parent class) va Engine (child**  **class), bunda Car obyekti Engine**  **obyektisiz mavjud bo’lmaydi.** |
| **Type of**  **Relationship** | **HAS-A munosabat** | **PART-OF munosabat** |
| **Type of association** | **Kuchsiz assotiativ (weak**  **association)** | **Kuchli assotiativ (strong**  **association)** |

**10. Obyektga yo’naltirilgan dasturlashda sinflar orasida vorislik tamoyili. Voris sinf yaratish. Vorisli turlari. Misollar keltiring**

**Javob:**

**Vorislik OYD (obyektga yo'naltirilgan dasturlash)**

**ning muhim tamoyili hisoblanadi.**

**Vorislik – bu dasturlash jarayonida bir sinf boshqasinfning xususiyatlarini (atributlari va usullarini) voris (meros) qilib olish imkonini beradigan mexanizmdir.**

**■ Asosiy tushunchalari:**

**– Super (base, parent) class (Asos sinf)**

**– Sub (derived, child) class (Voris sinf)**

**– Reusability (Qayta foydalanish imkoniyati)**

**Java tilida voris sinf yaratish**

**Java tilida voris sinf yaratish uchun extends kalit so’zidan foydalaniladi va quyidagicha ishlatiladi**

**class voris\_sinf extends super\_sinf{...}**

**Izoh: Voris sinf obyekti yaratilgan vaqtda asos (base) sinfdan obyekt yaratilmaydi. Voris sinf yaratilganda base sinfning jonstruktori avtomatik chaqiriladi, lekin bunda base sinfning obyekti yaratilmaydi**

**Vorislik turlari**

**■ Obyektga yo’naltirilgan dasturlashda quyidagi ko’rinishdagi vorislik turlari mavjud:**

**– Sodda vorislik**

**– Ko’p sathli vorislik**

**– Ierarxik vorislik**

**instanceof operatori**

**Yaratilgan voris sinf qaysi asos sinfga tegishli ekanligini tekshirish uchun instanceof operatoridan foydalaniladi.**

**Voris sinf imkoniyatlari**

**■ Voris sinf asos sinfning ochiq bo’lgan (private bo’lmagan) atributlari va metodlaridan foydalana**

**oladi.**

**■ Voris sinf ichida yangi atribut e’lon qilish mumkin.**

**■ Voris sinf ichida yangi va asos sinf ichida mavjud bo’lmagan metod yaratish mumkin**

**■ Voris sinf ichida asos sinf ichida yozilgan metodni qayta yozish (overriding) mumkin**

**■ Voris sinf ichida asos sinf konstruktorini super kalit so’zi orqali chaqirish mumkin**

**■ Odatda asos sinf metodlari va konstruktorini chaqirish uchun super kalit so’zi ishlatiladi**

**11. Obyektga yo’naltirilgan dasturlashda polymorphism tamoyili. Sinf ichida metodlarni qayta yuklash (method overloading). Misollar keltiring.**

**12. Obyektga yo’naltirilgan dasturlashda polymorphism tamoyili. Sinf ichida metodlarni qayta yozish (method overriding). Misollar keltiring.**

**13. Obyektga yo’naltirilgan dasturlashda abstraktlik tamoyili. Java muhitida abstrakt metod va abstrakt sinflar yaratish. Abstrakt sinf xususiyatlarini misollar asosida tushuntiring.**

**Javob:**

**Abstrakt sinflar (Abstraction)**

**Abstrakt sinflar – bu yaratiladigan sinflar uchun shablon sifatida ishlatiladigan va voris sinflar**

**uchun kerakli bo’lgan metodlarni o’z ichiga oladigan sinf hisoblanadi.**

**Abstrakt sinf eng kamida bitta abstrakt metoddan iborat bo’lishi kerak.**

**Abstrakt sinf mavhumlik xususiyatiga ega bo’lganligi uchun undan obyekt yaratib bo’lmaydi.**

**API (Application programming interface) va freymvorklar yaratish uchun**

**abstrakt sinflardan shablon sifatida foydalaniladi.**

**14. Abstrakt sinf va interfeys tushunchasi. Java muhitida abstrakt sinf va interfeysdan foydalanish, ularning o’zaro farqini tushuntiring. Misollar keltiring.**

**Javob:**

**Abstrakt sinflar (Abstraction)**

**Abstrakt sinflar – bu yaratiladigan sinflar uchun shablon sifatida ishlatiladigan va voris sinflar uchun kerakli bo’lgan metodlarni o’z ichiga oladigan sinf hisoblanadi.**

**Abstrakt sinf eng kamida bitta abstrakt metoddan iborat bo’lishi kerak.**

**Abstrakt sinf mavhumlik xususiyatiga ega bo’lganligi uchun undan obyekt yaratib bo’lmaydi.**

**API (Application programmwing interface) va freymvorklar yaratish uchun**

**abstrakt sinflardan shablon sifatida foydalaniladi.**

**Interface –static va final o’zgaruvchilar va methodlar yig’indisi. Qisqa qilib aytganda**

**prototiplar to’plami.**

**Agar class interface dan nasl olsa, interface ichidagi hamma methodlarni class da yozib**

**chiqishi kerak.**

**Interface e’lon qilish:**

**interface Math {**

**final int COUNT = 10;**

**static int MAX=100;**

**void salom\_math();**

**int sum(int a, int b);**

**}**

**15. Java muhitida packagelar bilan ishlash**

**Javob:**

**Java paket (package) – bu klasslarni ma’lum bir struktura asosida joylashtirish, yani klasslarni namespace larga bo’lib tashlash. Java packetlar JAR faylda ham bo’lishi mumkin. JAVA packetlar classlarning manosiga qarab bo’linadi.**

**Paketlar quyidagi toifalarni o’z ichiga oladi:**

**Classlar**

**Interfacelar**

**Enum toifalar**

**16. Java muhitida istisnolarni boshqarish. Try, catch, throw, throws, finally kalit so’zlaridan foydalanib istisnolarni boshqarish. Misollar keltiring**

**Javob:**

**Istisno holatlar**

**• Java tilida istisno — dastur kodi biror qismida hosil bo’lganistisno holatni tasvirlovchi ob’ektdir.**

**Istisno holat yuzaga kelgandaException sinfi ob’eykti yaratiladi. Bu obyekt shu tipdagi istisno**

**holatni qayta ishlovchi usulga jo’natadi. Istisnolar har xil holatlar haqida ma’lumot berish uchun**

**«qo’lda» yaratilishi mumkin.**

**• Java tilida istisnolarni qayta ishlash mexanizmi beshta kalit so’zdan foydalaniladi: — try, catch, throw, throws va finally. Bu mexanizm ishlashi quyidagicha.**

**• Siz kod blokini bajarishga urinasiz (try)**

**• Bunda xatolik yuzaga kelsa, tizim istisnoni (throw) generatsiya qiladi.**

**• Bu istisno tipiga qarab ushlab olish (catch) yoki ko’zda tutilgan qayta ishlovchiga (finally)**

**uzatishingiz mumkin.**

**Quyida istisnolar qayta ishlash bloki umumiy ko’rinishi keltirilgan.**

**try {**

**// kod bloki**

**}**

**catch (TypeException1 exp) {**

**// TypeException1 turidagi istisnoni qayta ishlovchi kod**

**}**

**catch (TypeException2 exp ) {**

**// TypeException2 turidagi istisnoni qayta ishlovchi kod**

**throw(exp) // istisnoni qayta yaratish**

**}**

**finally {**

**}**

**Istisnolar tiplari**

**Throwable sinfi – ierarxiyaning eng yuqarisida turgan sinf hisoblanadi. Throwable sinfining ikkita voris sinfi mavjud.**

**1. Ехception sinfi — foydalanuvchi dastur kodi tomonidan ushlab olinishi lozim bo’lgan istisnolarni**

**ta’riflash uchun ishlatiladi.**

**2. Error – sinfi foydalanuvchi dastur kodi tomonidan ushlab olinmasliki lozim bo’lgan istisnolarni**

**ta’riflash uchun ishlatiladi.**

**Ushlab olinmagan istisnolar:**

**class Exc0 {**

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

**int d = 0;**

**int a = 42 / d;**

**}**

**}**

**• Natija:**

**• С:\> java Exc0**

**• java.lang.ArithmeticException: / by zero at**

**Exc0.main(Exc0.java:4)**

**try va catch**

**Istisnolardan himoya qilish lozim bo’lgan dastur kodini berzish uchun try kalit so’zi ishlatiladi.**

**Bunda try – blokdan so’ng qayta ishlash lozim bo’lgan istisno tipini beruvchi catch blok joylashadi.**

**Misol:**

**class Exc2 {**

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

**try {**

**int d = 0;**

**int a = 42 / d;**

**}**

**catch (ArithmeticException e) {**

**System.out.println("division by zero");**

**}**

**}**

**}**

**Bir nechta catch bo’limlari**

**Ba’zi hollarda bitta dastur kodi turli tipdagi istisnolar yaratishi mumkin. Bunday holatlarni qayta**

**ishlash uchun, Java try – blok uchun ixtiyoriy catch – bloklar kiritishga imkon beradi. Quyidagi**

**dasturda ikkita tipdagi istisno ushlanadi.**

**class MultiCatch {**

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

**try {**

**int a = args.length;**

**System.out.println("a = " + a);**

**int b = 42 / a;**

**int c[] = { 1 };**

**c[42] = 99;**

**}**

**catch (ArithmeticException e) {**

**System.out.println("div by 0: " + e);**

**}**

**catch(ArrayIndexOutOfBoundsException e) {**

**System.out.println("array index oob: " + e);**

**}**

**}**

**}**

**Joylashtirilgan try операторлари**

**Xuddi o’zgaruvchilar ko’rinish sohalari kabi try operatorlarini bir biriga ichma – ich joylashtirish**

**mumkin. Agar quyi darajadagi try operatorida yaratilgan istisnoga mos catch bo’limi mavjud**

**bo’lmasa, tashqi try operatori catch bo’limlari tekshiriladi.**

**Misol:**

**class MultiNest {**

**static void procedure() {**

**try {**

**int c[] = { 1 };**

**c[42] = 99;**

**}**

**catch(ArrayIndexOutOfBoundsException e) {**

**System.out.println("array index oob: " + e);**

**}**

**}**

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

**try {**

**int a = args.length();**

**System.out.println("a = " + a);**

**procedure();**

**int b = 42 / a;**

**}**

**catch (ArithmeticException e) {**

**System.out.println("div by 0: " + e);**

**}**

**}**

**}**

**17. Java muhitida fayllar bilan ishlash. File sinfi va faylga ma’lumot yozish sinflaridan foydalnish (FileOutputStream, FileWriter)**

**Javob:**

**Faylga ma’lumot yozish**

**Bunda bir qancha metodlar mavjud.**

**Example:**

**Public void write (int c ) 🡪 Simvol yozadi**

**Public void write(String str , int off , int len) 🡪 String yozadi.**

**Public void write (char[] , int off , int len) 🡪 Massivdagi simvolni yozish.**

**Public void else() 🡪 oqimni yopish.**

**String massivni faylga yozish:**

**Import java.io.FileWriter;**

**Public class Main{**

**Public void static main(String[] args) throws(exption){**

**FileWriter fw =new FileWriter(“file.txt”);**

**String strs[]={“com”,”exe”,”doc”};**

**For(int i=0 ; i<strs.lenght ; i++){**

**Fw.write(strs[i] + “\n”);**

**}fw.close;**

**}**

**}**

**18. Java muhitida fayllar bilan ishlash. File sinfi va fayldan ma’lumot o’qish sinflaridan foydalnish (FileInputStream, FileReader, BufferedReader)**

**Javob:**

**Fayldan ma’lumot o’qish**

**Construction :**

1. **File Reader (File File);**
2. **File Reader (File);**
3. **File Reader (String file Name);**

**Method:**

1. **Int read;**
2. **Void close;**
3. **File Reader (String file Name);**

**Example:**

**Import java.io.\*;**

**Public class Read{**

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

**File Reader text = new FileReader(“asb.txt”);**

**Int inByte;**

**Do{**

**Int Byte = text.read();**

**If(inByte!=1);**

**System.out.print((char) inByte);}**

**While(inByte!=-1);**

**System.out.print(“\_”);**

**Text.close;**

**}**

**}**

**}**