Pravdivé odpovědi (plus některé nepravdivé označené X):

o Metoda drawRect(int x, int y, int width, int height) vykreslí obrys obdélníka.

o Třída Graphics používaná u komponent v metodě paint má instanční metodu setColor pro určení barvy, kterou se bude kreslit. Tato barva bude použita pro kreslení obrysů (metody drawXXX) i pro kreslení výplní (metody filXXX).

o Metoda drawArc(int x, int y, int width, int height, int startAngle, int arcAngle) umožňuje vykreslit oblouk. Jeho střed bude na pozici [x+width/2,y+height/2].

o Pokud chceme překreslit námi definovanou komponentu, zavoláme metodu repaint() této komponenty. Tuto metodu volá i systém v případě požadavku na překreslení komponenty (např. okno s komponentou se dostává do popředí).

o Nejpoužívanější databázové systémy využívají relační model. Dalšími databázovými modely jsou hierarchický či síťový.

o V Javě je IP adresa reprezentována třídou InetAddress. Pro jednoho hostitele nemusí být jednoznačně určena IP adresa a tak mezi její statické metody patří i metoda getAllByName(String host), jejímž výsledkem je pole typu InetAddress[], v kterém jsou uloženy všechny adresy hostitele dle DNS.

o URL se skládá z identifikátoru protokolu a jména zdroje. Jméno zdroje se dále může skládat z těchto částí //host:port/soubor#reference.

o Rozhraní Connection předepisuje metodu prepareStatement(...) pro vytvoření objektu typu PreparedStatement.

o Rozhraní Statement předepisuje metodu executeQuery(String sql) která vrací hodntou Result Set.

o Kód Statementu je vytvořen (a uložen) na klientovi.

o Konfigurabilita Statementu je vysoká.

o Efektivita přenosu dat Statementu je nízká.

o Kód Callable Statementu je vytvořen (a uložen) na serveru.

o Konfigurabilita Callable Statementu je nízká.

o Efektivita přenosu dat u Prepared Statementu je nízká při prvním použití, pak je vysoká.

X Kód Prepared Statementu je vytvořen na serveru.

o Tento kód

```
PreparedStatement ps = con.prepareStatement(
" UPDATE employees  SET salary = ? WHERE id = ? " );
ps.setBigDecimal( 1, "select * from salary WHERE 17" );
ps.setInt( 2, 777 );
```

nepůjde přeložit, protože metoda setBigDecimal očekává jako druhý parametr java.math.BigDecimal

o Metoda getConnection(...) třídy DriverManager se snaží najít vhodný driver pro připojení k databázi. Pokud se to nepodaří vyvolá výjimku typu SQLException.

o Metoda getConnection(...) třídy DriverManager je statická a snaží se najít vhodný driver pro připojení k databázi.

o Vizuální komponenty se měří v obrazových bodech (pixelech).

o Událost v GUI je objekt popisující jev a poskytující metody k obsluze.

o Tento kód:

```
g.setFont( new Font( "courier", Font.ITALIC + Font.BOLD, 18 ) );
g.drawString( "Hello, World",  70, 400 );
```

umožní vypsat řetězec Hello, World fontem velikosti 18 bodů.

o Při programovém volání metody repaint(), awt vlákno zavolá metodu update(Graphics g), která defaultně obsahuje:

```
public void update( Graphics g ) {
 g.clearRect(...);
paint( g );
 }
```

o Registrace ovladače JDBC provede metoda

```
void registerDriver( Driver driver )
```

. Tato metoda je definována ve třídě DriverManager.

o Pro spojení s relační databází se v Javě používá JDBC. Jiné řešení (zejména pokud JDBC není k dispozici) je použití ODBC (Open Database Connectivity). Pak je nutné využít JDBC-ODBC bridge.

o Pro přístup ke konkrétnímu databázovému serveru je potřeba JDBC ovladač(driver), který poskytuje tvůrce databázového serveru.

o Systémy řízení bází dat (DBMS) jsou velmi rozsáhlé softwarové systémy produkované specializovanými výrobci - např.: MySQL, Derby, Oracle, PointBase, Cloudscape, Sybase, IBM DB2.

o Při překladu javského programu připojujícímu se k databázi není nutné přesně vědět výrobce databáze, příslušný driver je možné načíst v závislosti na vstupních parametrech programu.

o Primární klíč v relačních databázích je jednoznačný identifikátor záznamu, řádku tabulky. Primárním klíčem může být jediný sloupec či kombinace více sloupců tak, aby byla zaručena jeho jednoznačnost.

o Systém řízení bází dat umožňuje data uchovávat, řadit a také umožňuje přístup uživatelů podle definovaných oprávnění.

o Příkaz

```
Class.forName( "oracle.jdbc.driver.OracleDriver" );
```

způsobí načtení třídy oracle.jdbc.driver.OracleDriver do JVM. Tato třída musí být dostupná z úložiště dostupného z CLASSPATH.

X Následující kód

OracleDriver od = new Driver(Class.forName("oracle.jdbc.driver.OracleDriver"));

vytvoří nový JDBC driver podle definice v oracle.jdbc.driver.OracleDriver.

o TreeSet je potomkem SortedSet.

o Novou kolekci s klíči typu String a hodnotami typu Item vytvoříme takto: Map <String, Item> map = new TreeMap <String, Item> ( );

o DNS (Domain Name System) je jmenná vyhledávací služba, která umožňuje převádět slovní zápis Internetové adresy (např. www.cvut.cz) na její číselnou podobu (147.32.3.39).

o URL se skládá z identifikátoru protokolu a jména zdroje. Identifikátorem může být např. http, ftp, file či jdbc.

o IP adresa je 32-bitová (resp. 128-bitová) identifikace síťového rozhraní v počítačové síti, které používá IP protokol.

o Třída implementující rozhraní Comparator musí definovat metodu public int compare(Object x, Object y), která vrací hodnotu nula pokud jsou objekty x a y stejné.

o Tento kód:

```
try {
        InetAddress a = InetAddress.getByName("cvut.cz");
        byte[] b = a.getAddress();
        System.out.println(b[0] + "." + b[1] + "." + b[2] + "." + b[3]);
} catch (UnknownHostException ex) {
ex.printStackTrace();
};
```

nevypíše správný výsledek pro cvut.cz (výsledek by měl být 147.32.3.39).

X Tento kód:

```
try {
        InetAddress a = InetAddress.getByName("cvut.cz");
        byte[] b = a.getAddress();
        System.out.println(b[0] + "." + b[1] + "." + b[2] + "." + b[3]);
} catch (UnknownHostException ex) {
ex.printStackTrace();
};
```

půjde přeložit, ale vyvolá výjimku UnknownHostException, protože Internetová adresa musí mít specifikovaný identifikátor protokolu (http, ftp,..).

X Port je zásuvka pro síťové spojení. Počet portů v počítači je dán počtem síťových zásuvek, jejich počet v jednom počítači nesmí překročit 1024 (číslovány jsou od 0 do 1023).

o Aby se ustavilo spojení mezi klientem a serverem, musí znát klient URL a číslo portu serveru.

Mějme následující definice tříd:

```
class Nit implements Runnable{
   int a = 0;
   static int b = 0;
   public void run() {
      for (int i = 0; i < 100; i++) {
         System.out.printf("i: %3d, a: %3d, b: %3d%n",i,++a,++b);
      }
   }
}
public class Main {
   public static void main(String[] args) {
      Nit n = new Nit();
      Thread t1 = new Thread(n), t2 = new Thread(n);
      t1.start();
      t2.start();
      System.out.println("Odstartovano");
   }
}
```

o Výpis by mohl začínat takto:

```
i:  0, a:  1, b:  1
i:  1, a:  2, b:  2
i:  2, a:  3, b:  3
```

o Výpis by mohl začínat takto (tedy posloupnost proměnné a může mít vynechanou 2):

```
Odstartovano
i:  0, a:  1, b:  1
i:  1, a:  3, b:  3
i:  2, a:  4, b:  4
```

o V metodě main jsou vytvořena a odstartována dvě vlákna.

o Jednou odstartované vlákno metodou start(), již nejde znovu odstartovat - tuto metodu nesmíme nad stejným objektem volat dvakrát.

o Pokud bychom před příkaz

```
System.out.println("Odstartovano");
```

přidali příkaz

```
t2.join();
```

bude muset hlavní vlákno počkat na dokončení vlákna t2, ale výpis nemusí být zakončen výpisem Odstartovano.

o Výpis bude mít včetně řádku pro text Odstartovano 201 řádek.

o Protože jsou obě vlákna t1 i t2 vytvořena nad stejným objektem typu Nit budou sdílet proměnnou a. Nejnižší hodnota proměnné a ve výpisu bude 1 a nejvyšší 200.

o Výpis může, ale nemusí končit řádkem: i: 99, a: 200, b: 200

o Metoda notify je definována u každého objektu a umožňuje aktuálně běžícímu vláknu uvolnit jedno náhodně vybrané vlákno uspané metodou wait volanou na tentýž objekt.

X Výpis může, ale nemusí končit řádkem: i: 99, a: 100, b: 100

X Před příkazem t1.start() je vlákno t1 ve stavu new (vytvořeno) a výraz t1.isAlive() má hodnotu true.

X Vlákno t1 musí skončit před koncem vlákna main, tj. vlákna, v kterém běží metoda main.

X Vlákno t2 je nastartováno ihned po dokončení běhu vlákna t1.s

X Protože jsou obě vlákna t1 i t2 vytvořena nad stejným objektem typu Nit budou sdílet proměnnou i. Nejnižší hodnota proměnné i ve výpisu bude 0 a nejvyšší 99.

X Vlákno t2 je nastartováno ihned po dokončení běhu vlákna t1.

X Výpis by mohl začínat takto:

i: 1, a: 3, b: 3
i: 2, a: 4, b: 4

X Vlákno t1 musí skončit před koncem vlákna main, tj. vlákna, v kterém běží metoda main.

X Celý výpis bude v jednom řádku, protože metoda printf neodřádkovává správně. Pokud bychom chtěli výpis do sloupečku, museli bychom použít metodu println nebo odřádkování pomocí sekvence \n.

Mějme následující Java kód:

```
1. public class A extends B{
2. { System.out.println("2");}
3. static{ System.out.println("5");}
4. A(){
5. System.out.println("T");
6. }
7. public static void main(String[] args) {
8. A d = new A();
9. B c = new B();
10. }
11. }
12. class B{
13. { System.out.println("8");}
14. static{ System.out.println("4");}
15. B(){
16. System.out.println("F");
17. }
18. }
```

Předpokládejte spuštění metody main. Vyberte pravdivé výroky.

o Ve výpisu bude číslice 2 právě jednou.

o První bude ve výpisu číslice 4.

o Ve výpisu bude číslice 4 právě jednou.

o Před voláním konstruktoru třídy musí být dokončen statický inicializátor třídy.

o Přesně před každým výpisem F je vypsána číslice 8.

X Statický inicializátor je volán vždy, když je vytvářen nový objekt.

X Ve výpisu bude číslice 3 právě jednou.

X V každé třídě musí být alespoň jeden nestatický inicializátor.

X Pokud je v definici třídy více nestatických inicializátorů, pak jsou volány v libovolném pořadí.

X První bude ve výpisu písmeno T.

Mějme následující definice tříd:

```
public class A extends B{
public int a = 2;
A(int i){ a=i;}
}

public class B{
public int b = 11;
B(){b=8;}
B(int a){b=2*a;};
}
```

a následující deklarace a definice:

```
B[] b = new B[2];
b[0] = new B(4);
b[1] = new B(2);
A[] a = new A[4];
B[][] bb = {{new B(), new B(2)},{new B()},{new B(5), new B(13), new B(24)}};
A[][] aa;
```

o Toto je správně zapsaný příkaz: b=a;

o Pole b seřadíme příkazem Arrays.sort(b);, pouze pokud by třída B implementovala rozhraní Comparable.

o Toto je správně zapsaný příkaz: a[1]=new A(123456789);

X Toto je správně zapsaný příkaz: a=b;

X Příkaz System.out.println(b.getClass( )); vypíše řetězec začínající Array [L.

X Toto
bb[1][1].b
je správně zapsaný výraz a jeho hodnotou je 8.

Zadání otázky
Mějme následující deklarace:
int[] p = {2,6,7};
int[][]p2 = new int[5][];
int [][][][] p3 = new int[3][5][3][5];
byte[] b = new byte[3];
Které následující příkazy mohou být použity? Každou odpověď posuzujte bez ohledu na ostatní.

o p[1] = 456;

o p = new int[] {2,4,6,7};

o p3[1][1][1][1] = b[2];

X p[3] = 100000;

X p3[2,3] = p2;

X p2[1] = 23;